Unterschrift des Betreuers

TECHNISCHE UNIVERSITÄT WIEN
Vienna University of Technology

DIPLOMARBEIT

Study of potential improvements of the CMS $H \to \tau^+\tau^-$ analysis using artificial neural networks with multiple layers

AUSGEFÜHRT AM

Atominstitut Wien
in Verbindung mit dem Institut für Hochenergiephysik (HEPHY)
der Österreichischen Akademie der Wissenschaften (ÖAW)

UNTER DER ANLEITUNG VON

Univ.Doz. Dr.techn. Rudolf Frühwirth
Dr. Martin Flechl

EINGEREICHT AN

der Technischen Universität Wien
Fakultät für Physik

VON

Spanring Markus
Hauslehen 78
3342, Opponitz, Österreich

Ort, Datum

Unterschrift (Student)

Abstract

After the discovery of a boson with a mass of approximately 125 GeV in 2012, the next step is to test whether this boson is the Standard Model Higgs boson, e.g. by measuring its couplings to fermions and gauge bosons. One of the fermion couplings is measured by looking at the Higgs $\rightarrow \tau^+\tau^-$ decay channel. One way to reach the traditional discovery significance of $5\,\sigma$ for this decay is to increase the amount of data. Another way is to find a good signal-versus-background classifier to extract the desired events from the recorded data more efficiently. To obtain this classifier a standard way is to use machine learning models, for instance neural networks. Recent advances in the field of deep learning have shown that deep neural networks are able to retrieve more information out of a given set of input functions than other machine learning methods. The main goal of this thesis is to use deep-learning techniques on a simulated Higgs $\rightarrow \tau^+\tau^-$ dataset and to compare the performance with other current Machine Learning (ML) techniques. The training is performed with a GPU-accelerated python library. For the tuning of the hyperparameters, a Bayesian optimization algorithm is used. The obtained result is that deep neural networks trained on this simulated dataset can not compete with the other ML techniques used as a benchmark. A possible explanation is that the training set is by far too small to train the deep neural network at a competitive level.

Kurzzusammenfassung

Nachdem ein Boson mit einer Masse von ungefähr 125 GeV im Jahr 2012 entdeckt wurde, ist der nächste Schritt zu beweisen, dass es sich dabei um das Standardmodell-Higgs-Boson handelt. Dies kann erreicht werden indem man z.B. die Kopplungen dieses Bosons zu Fermionen misst. Eine dieser Kopplungen kann durch die Messung des Zerfalls Higgs $\rightarrow \tau^+\tau^-$ erfolgen. Eine Möglichkeit um die in der Teilchenphysik übliche Signifikanz von $5\,\sigma$ für diesen Zerfall zu erreichen, ist die Menge an Daten zu erhöhen. Eine weitere Möglichkeit ist es, eine geeignete Methode zu finden mit welcher dieses Signal effizienter von Hintergrundereignissen unterschieden werden kann. Eine übliche Herangehensweise an ein solches Problem ist der Einsatz von maschinellem Lernen (ML) wie zum Beispiel neuronaler Netze. Der Fortschritt im Bereich von "Deep Learning", unter anderem bei vielschichtigen neuronalen Netzwerken, zeigte, dass diese Art von Netzwerken deutlich mehr Information aus einem Datensatz extrahieren kann als andere im maschinellem Lernen eingesetzte Methoden. Die Hauptaufgabe dieser Arbeit besteht darin, solche vielschichtigen neuronalen Netze mithilfe eines simulierten Higgs $\rightarrow \tau^+\tau^-$ Datensatzes zu trainieren und mit anderen ML-Methoden zu vergleichen. Hierfür wird eine GPU-beschleunigte Python-Bibliothek verwendet. Um die optimalen Werte der Hyperparameter des neuronalen Netzwerkes zu finden, wurde ein Bayesscher Optimierungsalgorithmus verwendet. Daraus resultierend konnte festgestellt werden, dass dieses Netzwerk, welches mit dem simulierten Datensatz trainiert wurde, nicht mit anderen ML-Methoden konkurrieren kann. Ein möglicher Grund ist, dass der verwendete Datensatz für die Anwendung von vielschichtigen neuronalen Netzwerken zu klein ist um diese auf ein konkurrenzfähiges Niveau zu trainieren.

Table of Contents

1 Introduction

The Higgs mechanism was first introduced in 1960 to explain massive gauge bosons and how particles such as fermions and massive gauge bosons gain mass. The discovery of the massive W- and Z-bosons, which are postulated by the Higgs mechanism, was a strong indication that this theory is correct. The Standard Model was tested with various precision measurements and continuously verified, e.g. by the discovery of the top quark in 1994, during the last decades. In 2012 the discovery of a boson with a mass of 125 GeV as well as spin and CP properties compatible with the Higgs boson was announced by CERN. To identify this boson as the Standard Model Higgs boson, among other things, its couplings to fermions, which are generated via Yukawa interaction, need to be confirmed. One suitable candidate for this task is the decay of this boson to a tau-antitau lepton pair (Higgs$\rightarrow \tau^+\tau^-$).

In this thesis a study is conducted on whether a deep neural network is able to enhance the performance of extracting Higgs$\rightarrow \tau^+\tau^-$ events from a collected dataset. For this purpose such a deep neural network is trained and compared to other Machine Learning (ML) approaches as well as to a cut-based approach which is the standard analysis technique in high energy physics. Since deep neural networks are a relatively new field of study, especially in high energy physics, only the framework on which a deep neural network can be constructed is available. The actual software as well as a suitable hardware is built from scratch.

First, a short overview of the Standard Model of particle physics is given, with the emphasis on spontaneous symmetry breaking and the Higgs Boson (Chap. 2). Thereafter, the *Large Hadron Collider* (LHC) at CERN is introduced with focus on the CMS detector and its subdetectors (Chap. 3). Chapter 4 deals with the theoretical framework of an artificial neural network including training algorithms and properties of the network architecture as well as the definition of a deep neural network. The following chapter (Chap. 5) outlines the software and hardware as well as the datasets used to construct and train a deep neural network and introduces the *Figure of Merit* (FoM) used to evaluate the performance of the neural network. In chapter 6 a simulated dataset for 8 TeV corresponding to an integrated luminosity of $\mathcal{L} = 19.7\,\mathrm{fb}^{-1}$ is used to train and optimize a deep neural network. Two optimization approaches are put into practice and compared with each other. The last optimization step for the simulated 8 TeV dataset is the reduction of the input variables to a smaller, better performing subset and the application of preprocessing on these input variables. In chapter 7 a simulated dataset for 13 TeV corresponding to an integrated luminosity of $\mathcal{L} = 10\,\mathrm{fb}^{-1}$ is used.

For 13 TeV, significantly more simulated events are available than in the 8 TeV dataset and can in addition be used to study the influence of the dataset size on the performance of a deep neural network. Finally the results obtained with the 8 TeV and 13 TeV dataset are discussed and compared with other ML approaches in Chap. 8.

Throughout this thesis the particle mass is given in natural units (the units are chosen such that $c = \hbar = 1$) and therefore in *electron Volt* (eV). The charge is given in units of the *elementary charge* e_0.

2 The Standard Model of particle physics

The *Standard Model of particle physics* (SM) is the theory of fundamental particles and their interactions. These interactions are responsible for the electromagnetic, weak and the strong force. Despite the fact that special relativity is incorporated in this theory, general relativity is not. Thus, the fourth fundamental force, gravitation, is not included in this theory. Although the SM describes all its particles and the resulting phenomenons nearly perfectly, it still has a lot of shortcomings some of which are listed below:

1. Why are there exactly 3 quark and lepton generations?
2. How can dark matter and dark energy be explained?
3. Is it possible to combine the electromagnetic, weak and the strong force to a *Grand Unified Theory* (GUT)?

The fundamental particles are divided into quarks, leptons, gauge bosons and the Higgs boson. Whereas quarks and leptons are the building blocks of matter, gauge bosons are the mediator particles of the forces. In addition, all of them have a counter particle called anti-particle. Anti-particles have the same physical properties except for the opposite sign of the charge. In the following sections the properties of these particles as well as their interactions are outlined.

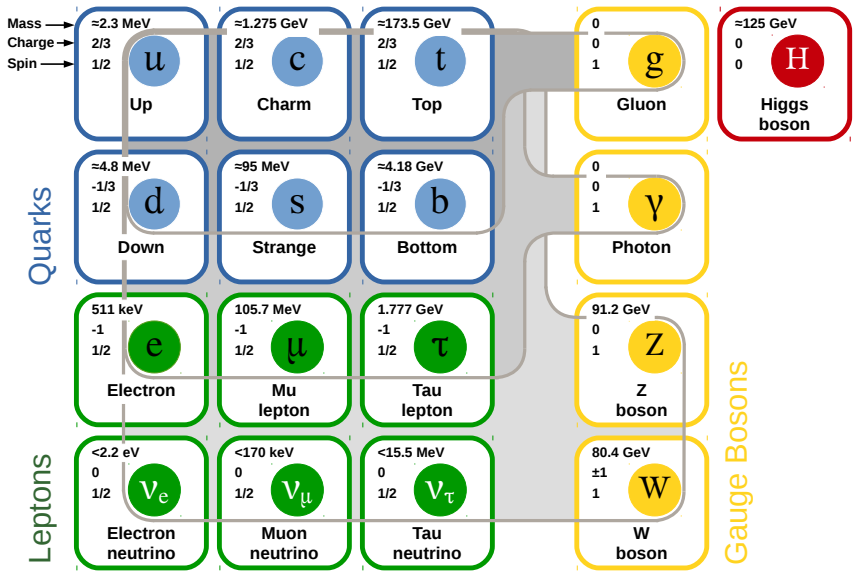


Fig. 2.1: The particles of the Standard Model.

2.1 Particles of the Standard Model

2.1.1 Quarks

A quark is an elementary particle with spin 1/2 (fermion) which can only be observed with other quarks in a composite particle called "hadron". The reason for not observing isolated quarks will be given in Section 2.2.3. Quarks come in six different types (also called flavors) as shown in Table 2.1. There are two kinds of hadrons which can be subdivided into particles with three quarks called baryons, and into particles with a quark-antiquark pair called mesons. Besides the electric charge, quarks also have a color charge (Red, Blue, Green and the corresponding anticolor). Quarks are the only particles in the Standard Model which interact via all four fundamental interactions (see Section 2.2). Out of the many particles consisting of quarks the proton and the neutron are the ones with the longest lifetime: The proton is believed to be stable, and the neutron decays (if it is not in a bound state) with a mean lifetime of about 15 min. All other hadrons decay significantly faster.

Name	Mass	Charge	Symbol
Up	$2.3^{+0.7}_{-0.5}\,\mathrm{MeV}$	$+\frac{2}{3}$	u
Down	$4.8^{+0.7}_{-0.3}\,\mathrm{MeV}$	$-\frac{1}{3}$	d
Charm	$1.275 \pm 0.025\,\mathrm{GeV}$	$+\frac{2}{3}$	c
Strange	$95 \pm 5\,\mathrm{MeV}$	$-\frac{1}{3}$	s
Top	$173.5 \pm 0.6 \pm 0.8\,\mathrm{GeV}$	$+\frac{2}{3}$	t
Bottom	$4.18 \pm 0.03\,\mathrm{GeV}$ $\left(\overline{MS}\right)$[1]	$-\frac{1}{3}$	b

Tab. 2.1: Properties of the different quark flavors [1].

2.1.2 Leptons

Like quarks, leptons are elementary particles with spin 1/2 and therefore fermions. Leptons can be subdivided further into three families, each consisting of a charged and a neutral particle. The best known representative of the charged leptons is the electron, which together with quarks makes up all visible matter in the universe. The neutral leptons are called neutrinos.
Charged leptons interact via the weak, electromagnetic and gravitational force. Neutrinos interact only weakly and gravitationally since they have no electric charge. Since neutrinos and anti-neutrinos have no charge, it is possible that the neutrino is its own anti-particle.

[1] The minimal subtraction scheme or MS-scheme was introduced by t'Hooft [2] in 1972 which is a renormalization scheme that only absorbs the divergent part of perturbative calculations beyond leading order. The more widely used $\overline{MS}$-scheme [3] absorbs the divergent part and also a universal constant.

Name	Mass	Charge	Symbol
Electron	511 keV	−1	e
Electron neutrino	< 2.2 eV	0	ν_e
Mu lepton	105.7 MeV	−1	μ
Muon neutrino	< 170 keV	0	ν_μ
Tau lepton	1.777 GeV	−1	τ
Tau neutrino	< 15.5 MeV	0	ν_τ

Tab. 2.2: Properties of the different lepton flavors [1].

Since the tau lepton is the most important particle in this thesis (besides the Higgs) it is discussed separately.

Tau lepton

The tau lepton is the heaviest member of the lepton family, as one can see in Tab. 2.2, and has a mean lifetime of about $2.9 \cdot 10^{-13}$ s [1]. The tau lepton was first observed in electron-positron[2] collision experiments from 1974 to 1977 at Stanford [4]. In these experiments event signatures of the form

$$e^+ + e^- \xrightarrow{?} e^\pm + \mu^\mp + \text{at least two undetected particles} \tag{2.1}$$

were found. This was explained by the production of two tau leptons during the e^-e^+-collision which decay to an electron or mu lepton and four neutrinos via weak interaction.

$$e^+ + e^- \xrightarrow{collision} \tau^+ + \tau^- \xrightarrow{decay} e^\pm + \mu^\mp + 2\overline{\nu} + 2\nu \tag{2.2}$$

Since the tau lepton is heavier than some hadrons it can decay into a hadron and a tau neutrino. About $^2/_3$ of the decays are hadronic and $^1/_3$ decays either into an electron and two neutrinos or a muon and two neutrinos. The fact that the tau can decay both leptonically and hadronically plays an important role in collider physics because in contrast to electrons and muons hadronic taus are more difficult to reconstruct. For simplicity $\tau^+ + \tau^-$ is denoted as $\tau\tau$ from now on.

2.1.3 Gauge bosons

Gauge bosons are the carrier particles of the electromagnetic, weak and strong interaction. These interactions, described by so-called gauge theories, define how elementary particles interact with each other. The gauge boson of the fourth fundamental interaction, gravitation, is only hypothetical and has not been found yet. Forces acting between different particles are treated as the exchange of gauge bosons. The strength of these forces is determined by so-called coupling constants. These constants arise naturally in the quantum field theory describing the respective interaction. Usually,

[2]The positron is the anti-particle of the electron

Interaction	Name	Mass	Charge	Spin	Coupling	Symbol
Electromagnetism	Photon	0	0	1	$\frac{1}{137}$	γ
Weak	$W^{\pm}$-Boson	80.385 GeV	± 1	1	$\sim 10^{-6}$	$W^{\pm}$
	Z-Boson	91.188 GeV	0	1		Z
Strong	8 Gluons	0	0	1	1	g
Hypothetical particle						
Gravitation	Graviton	0?	0	2?	$\sim 10^{-39}$	G

Tab. 2.3: Properties of the three experimentally confirmed gauge bosons and the hypothetical gauge boson of gravitation [1, 6].

the formalism used to describe such a theory (e.g. Lagrange formalism [5]) can be separated into a kinetic and an interaction part. The couplings determine the strength of the interaction part with respect to the kinetic part. The coupling constants in Tab. 2.3 are given with respect to the coupling constant of the strong interaction.

2.2 The four fundamental interactions

2.2.1 Electromagnetism

Besides gravitation, nearly all phenomena of everyday life are based on electromagnetism. This includes all chemical phenomena and all forms of light.
The first complete classical theory of electromagnetism was given by James Clerk Maxwell [7]. He was able to unify electricity, magnetism and light into one theory, resulting in the Maxwell equations:

$$\partial_i E^i = \frac{\rho}{\epsilon_0}, \tag{2.3}$$

$$\epsilon_{ijk}\partial_j E^k - \dot{B}^i = 0, \tag{2.4}$$

$$\partial_i B^i = 0, \tag{2.5}$$

$$\epsilon_{ijk}\partial_j B^k - \epsilon_0\mu_0\dot{E}^i = \mu_0 J^i, \tag{2.6}$$

where E^i is the elctric field, B^i is the magnetic field, ρ is the charge density and J^i is the current density. The electromagnetic field described by this equations exerts the so-called Lorentz force $F^i_{Lorentz}$ on a charged particle q:

$$F^i_{Lorentz} = qE^i + q\epsilon_{ijk}v^j B^k. \tag{2.7}$$

These equations also imply that the speed of light in vacuum is constant.
In Maxwell's theory light is treated as a wave and propagates at a constant speed, an assumption which has led to the discovery of special relativity by Albert Einstein [8] at the beginning of the 20th century. This view had to be revised after the discovery of the photoelectric effect [9]. Paul Dirac was the first to describe the interaction

between radiation and matter by using quantum theory [10]. Based on this work it was possible to develop a theory of quantum electrodynamics which is a relativistic quantum field theory. It is the first theory fully consistent with special relativity and has served as template for all future quantum field theories such as for the weak and strong interaction. Quantum electrodynamics explains the electromagnetic force between charged particles through the exchange of a photon.

2.2.2 Weak interaction

The weak interaction acts on all known left-handed fermions. It is responsible for radioactive decay and nuclear fusion and is the only interaction which is able to change flavors of quarks (see Sec. 2.1.1). It is also the only interaction which does not produce bound states like gravitation, electromagnetism and the strong interaction. The weak nuclear force is caused by the emission and absorption of $W^{\pm}$- and Z-bosons. These three bosons have a mass much larger than the mass of the proton, and the W-bosons also carry an electric charge. The weak interaction has a very short range of about 10^{-17} m due to the heavy mass and therefore short lifetime of the mediator particles which is below 10^{-24} s. At low energies the Standard Model treats the electromagnetic and the weak interaction as two different forces. Glashow, Weinberg and Salam [11, 12] were able to show that these interactions are only two aspects of the more general electroweak interaction. The electroweak interaction is a $SU(2) \times U(1)$ gauge group with four massless gauge bosons. At low energies spontaneous symmetry breaking (see Sec. 2.3.2) of the $SU(2)$ symmetry results in three massive ($W^{\pm}$, Z) and one massless boson (photon). This breaking of the electroweak symmetry is caused by the Higgs mechanism [13]. Furthermore, the weak interaction is the only fundamental interaction which breaks parity- and also charge-parity-symmetry [14].

2.2.3 Strong interaction

The strong interaction is responsible for the binding force between quarks and gluons. The theory describing this interaction is quantum chromodynamics, which is based on a $SU(3)$ symmetry group. The strong force is mediated by eight gluons and acts on particles with color charge. A composite of such particles, held together by the strong force, is called a hadron. A hadron must have zero total color charge because of a phenomenon called confinement. Confinement states that color-charged particles like quarks are only observable in their bound states and not as isolated particles. The explanation is that the energy needed to separate two quarks is high enough to create a new pair of quarks. The attempt to separate two quarks therefore results in the production of new hadrons. The strong interaction is furthermore responsible for the binding force between nucleons (e.g. proton and neutron). Since nucleons are hadrons and do not have a color charge, no attractive force should be present between two of them. However, the gluon-field inside the nucleus is not completely saturated which leads to a fluctuating color charge. This remnant causes a nuclear force between nuclei which is mediated through an exchange of a virtual pion.

2.2.4 Gravitation

Gravitation is an attractive force between two masses m_1 and m_2 and can be written in the form

$$F^i = G \cdot \frac{m_1 \cdot m_2}{r^2} e_r^i, \tag{2.8}$$

where G stands for the gravitational constant and r is the distance between the two masses m_1 and m_2. The vector e_r^i is the unit vector in radial direction. Gravitation is the weakest of all four forces when described in terms of coupling constants (see Sec. 2.1.3). This means that gravitation can be neglected when describing the behaviour of subatomic particles. This is the reason why the Standard Model works so well on a microscopic scale even though it does not include all four forces. On a macroscopic scale however, gravitation becomes the dominant force. In a possible quantum field theory of gravitation the force is mediated through the Graviton (see Tab. 2.3). However, this particle is only hypothetical and has not been discovered so far.

2.3 Higgs mechanism

The Higgs mechanism [13] introduces the concept of spontaneous symmetry breaking and is used to explain why the gauge bosons of the weak interaction are massive. Furthermore, it is possible to show that the breaking of the $SU(2)$ symmetry of the elektroweak interaction leads to the electromagnetic and weak interaction with their respective gauge bosons. The following description of the Higgs mechanism is derived according to Cottingham [5].

2.3.1 Global symmetry breaking and Goldstone bosons

For a complex scalar field $\Phi = \left(\frac{1}{\sqrt{2}}\right) \cdot (\phi_1 + i\phi_2)$ a possible Lagrangian density is

$$\mathcal{L} = \left(\partial_\mu \Phi^\dagger\right) \partial^\mu \Phi - m^2 \Phi^\dagger \Phi, \tag{2.9}$$

with $\partial_\mu = \partial_t - \nabla$. The term $\left(\partial_t \Phi^\dagger\right)(\partial_t \Phi)$ can be regarded as the kinetic energy and $\left(\nabla \Phi^\dagger\right) \cdot \nabla \Phi + m^2 \Phi^\dagger \Phi$ as the potential energy. If Φ is a constant field in space and time the derivatives vanish. The only contribution to the energy is $m^2 \Phi^\dagger \Phi$. Since m^2 is positive the energy will have a minimum at $\phi_1 = \phi_2 = 0$, meaning that $\Phi = 0$ corresponds to the 'vacuum' state. One can now modify the Lagrangian density by introducing a potential term $V(\Phi^\dagger \Phi)$,

$$V(\Phi^\dagger \Phi) = \frac{m^2}{2\phi_0^2} \left[\Phi^\dagger \Phi - \phi_0^2\right]^2, \tag{2.10}$$

where ϕ_0^2 is a real parameter. The form of the potential $V(\Phi^\dagger \Phi)$ is shown in Fig. 2.2. The obtained Lagrangian is given by

$$\mathcal{L} = \left(\partial_\mu \Phi^\dagger\right) \partial^\mu \Phi - V(\Phi^\dagger \Phi), \tag{2.11}$$

and does have a ground state at $\Phi^\dagger \Phi = |\Phi|^2 = \phi_0^2$. This ground state is not unique

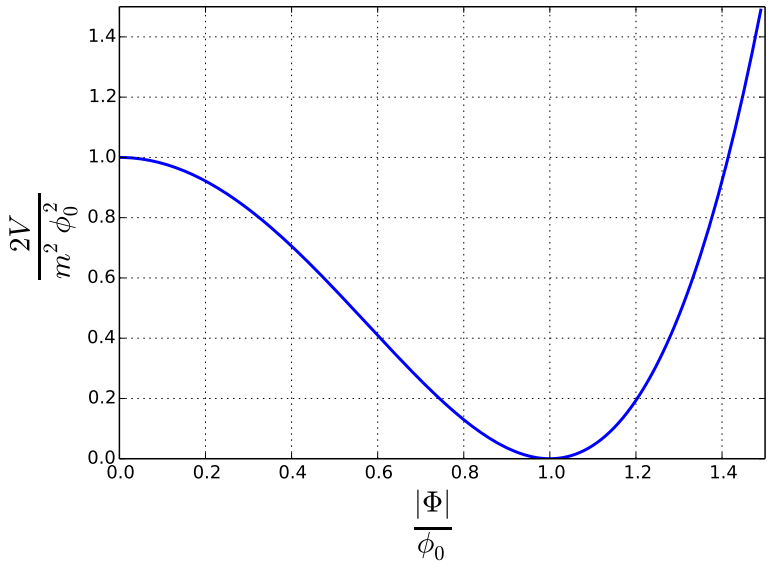


Fig. 2.2: Mexican hat potential $V(\Phi^\dagger\Phi)$ where Φ is a classical scalar field.

since there exists an infinite number of solutions, defined by a point on the circle $|\Phi| = \phi_0 = \sqrt{\phi_1^2 + \phi_2^2}$. Even though the Lagrangian density is invariant to rotations in the state space (ϕ_1, ϕ_2) the ground state does not show such symmetries. A similar example is the occurrence of ferromagnetism where the corresponding Hamiltonian may have rotational symmetry, but in its ground state it is magnetized in some particular direction and the rotational symmetry is lost. This is an example for spontaneous symmetry breaking. The Lagrangian has a $U(1)$-symmetry:

$$\Phi \to \Phi' = e^{-i\alpha}\Phi \tag{2.12}$$

where α is a real number. This transformation can also be written as:

$$\phi_1' = +\phi_1 \cos\alpha + \phi_2 \sin\alpha \tag{2.13}$$

$$\phi_2' = -\phi_1 \sin\alpha + \phi_2 \cos\alpha \tag{2.14}$$

This is a rotation in the state space (ϕ_1, ϕ_2), where $|\Phi|^2$ is constant. If one is looking at the real ground state $(\phi_0, 0)$, the $U(1)$ symmetry is broken. It is now possible to expand the Lagrangian density around this ground state introducing the following notation of Φ

$$\Phi = \phi_0 + \sqrt{\frac{1}{2}}(\chi + i\psi). \tag{2.15}$$

We get for the Lagrangian density

$$\mathcal{L} = \frac{1}{2}\partial_\mu\chi\partial^\mu\chi + \frac{1}{2}\partial_\mu\psi\partial^\mu\psi - \frac{m^2}{2\psi_0^2}\left[\sqrt{2}\phi_0\chi + \frac{1}{2}\chi^2 + \frac{1}{2}\psi^2\right]^2. \tag{2.16}$$

In place of the complex scalar field there are now two coupled scalar real fields. The Lagrangian density can now be divided into a free and an interaction part

$$\mathcal{L} = \mathcal{L}_{free} + \mathcal{L}_{int}, \tag{2.17}$$

with

$$\mathcal{L}_{free} = \frac{1}{2}\left[\partial_\mu\chi\partial^\mu\chi - 2m^2\chi^2\right] + \frac{1}{2}\partial_\mu\psi\partial^\mu\psi. \tag{2.18}$$

yielding a scalar field χ with mass $\sqrt{2}m$ and a massless scalar field ψ. The remaining part of the Lagrangian density contains the interactions between free particles and higher order corrections to their motion

$$\mathcal{L}_{int} = -\frac{2m^2}{8\phi_0^2}\chi\psi^2\left[\chi - 2\sqrt{2}\phi_0\right] + \frac{2m^2}{16\phi_0^2}\left[\chi^4 + 4\sqrt{2}\phi_0\chi^3 + \psi^4\right]. \tag{2.19}$$

The field ψ corresponds to a spin-0 particle with vanishing mass. Such massless fields arise as a result of global symmetry breaking and are called Goldstone bosons [15].

2.3.2 Local symmetry breaking and the Higgs boson

The next step is to construct a Lagrangian density that is invariant under a local $U(1)$ gauge transformation.

$$\mathcal{L} = \left[(\partial_\mu - iqA_\mu)\Phi^\dagger\right]\left[(\partial^\mu + iqA^\mu)\,\Phi\right] - \frac{1}{4}F_{\mu\nu}F^{\mu\nu} - V(\Phi^\dagger\Phi), \tag{2.20}$$

where $F_{\mu\nu} = \partial_\mu A_\nu - \partial_\nu A_\mu$ is the electromagnetic field strength tensor, q is the electromagnetic coupling constant and $V(\Phi^\dagger\Phi)$ the potential from eq. 2.10. The Lagrangian density $\mathcal{L}$ is invariant under local gauge transformation for a vanishing field A_μ and a constant field Φ defined by a point on the circle $|\Phi| = \phi_0$.

$$\Phi(x) \to \Phi'(x) = e^{-iq\theta}\Phi(x) \tag{2.21}$$

$$A_\mu(x) \to A'_\mu = A_\mu(x) + \partial_\mu\theta(x) \tag{2.22}$$

This again yields an infinite number of vacuum states. With a given Φ, $\theta(x)$ can always be chosen such that $\Phi'(x) = e^{-iq\theta(x)}\Phi(x)$ is real. When $\theta(x)$ is fixed, symmetry is broken since one is not free to make further gauge transformations. As already shown in Sec. 2.3.1, the potential Φ can be developed around the vacuum state ϕ_0

$$\Phi'(x) = \phi_0 + \frac{1}{\sqrt{2}}h(x), \tag{2.23}$$

where $h(x)$ is real. Inserting into the Lagrangian density yields,

$$\mathcal{L} = \left[(\partial_\mu - iqA'_\mu)(\phi_0 + \frac{1}{\sqrt{2}}h(x))\right]\left[(\partial^\mu + iqA'^\mu)(\phi_0 + \frac{1}{\sqrt{2}}h(x))\right]$$

$$-\frac{1}{4}F'_{\mu\nu}F'^{\mu\nu} - \frac{m^2}{2\phi_0^2}\left[\sqrt{2}\phi_0 h + \frac{1}{2}h^2\right]^2. \tag{2.24}$$

This Lagrangian density can now again be separated into a free and an interaction part $\mathcal{L} = \mathcal{L}_{free} + \mathcal{L}_{int}$ with

$$\mathcal{L}_{free} = \frac{1}{2}\left[\partial_\mu h \partial^\mu h - 2m^2 h^2\right] - \frac{1}{4} F_{\mu\nu} F^{\mu\nu} + q^2 \phi_0^2 A_\mu A^\mu, \tag{2.25}$$

$$\mathcal{L}_{int} = q^2 A_\mu A^\mu \left(\sqrt{2}\phi_0 h + \frac{1}{2}h^2\right) - \frac{m^2 h^2}{2\phi_0^2}\left(\sqrt{2}\phi_0 h + \frac{1}{4}h^2\right). \tag{2.26}$$

The free Lagrangian density describes a scalar boson field $h(x)$ of mass $\sqrt{2}m$ and a vector field A_μ which is equivalent to a vector boson of mass $\sqrt{2}q\phi_0$ with three independent components. This mechanism leads to massive bosons and was introduced amongst others by Peter Higgs [13]. The field $h(x)$ is called Higgs field. When the $SU(2)$ symmetry of the electroweak $SU(2) \times U(1)$ gauge group is broken one obtains a Lagrangian describing the three massive gauge bosons ($W^\pm$, Z) of the weak interaction, a massless photon field A^μ and the massive Higgs boson h:

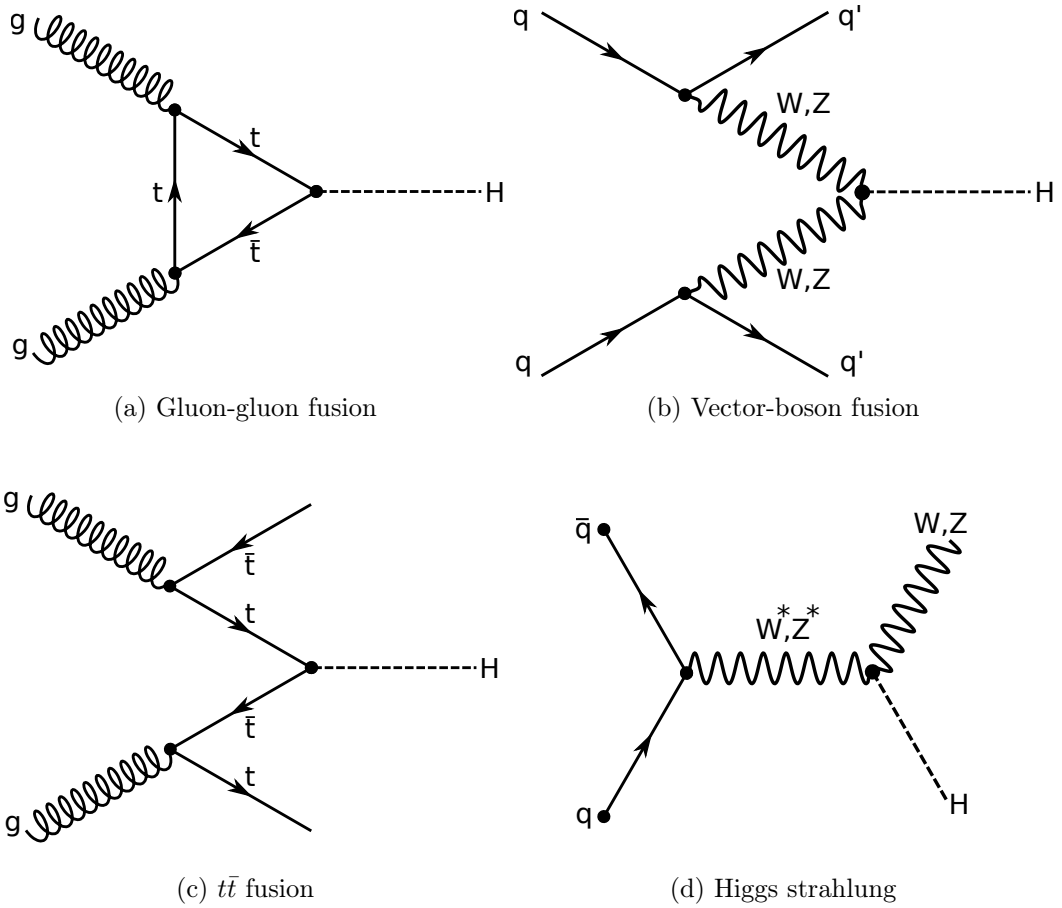
$$\begin{aligned}
\mathcal{L} = & \frac{1}{2}\left(\partial_\mu h \partial^\mu h - M_H^2 h^2\right) \\
& - \frac{1}{4} Z_{\mu\nu} Z^{\mu\nu} + \frac{1}{2} M_Z^2 Z_\mu Z^\mu \\
& - \frac{1}{4} A_{\mu\nu} A^{\mu\nu} \\
& - \frac{1}{2}\left[\left(D_\mu W_\nu^+\right)^* - \left(D_\nu W_\mu^+\right)^*\right]\left[D_\mu W_\nu^+ - D^\nu W^{+\mu}\right] + M_W^2 W_\mu^+ W^{-\mu}
\end{aligned} \tag{2.27}$$

2.3.3 Higgs boson production modes

Since the colliding particles at the LHC are hadrons, which are bound together by gluons, the dominating Higgs boson production mode is via *gluon-gluon-Fusion* (ggF)(see Fig. 2.3a). The Higgs boson does not couple directly to gluons; instead, the coupling is achieved through a top quark loop. The second-most-common production mode at the LHC is via *Vector Boson Fusion* (VBF)(Fig. 2.3b). In VBF, two quarks are each radiating a $W^\pm$ or Z^0 boson which combine to produce a Higgs boson. VBF events have a unique signature of two well-separated hadronic jets with large η gap (with the pseudo-rapidity $\eta = \ln\left[\tan\left(\theta/2\right)\right]$, where θ is the angle between particle and beam axis). Due to this clear signature the VBF events are studied in this thesis. The two remaining production modes are the least likely ones at the LHC (Fig. 2.3c and 2.3d). They are called associated production modes. There, the Higgs boson is either produced together with a $t\bar{t}$-pair or radiated off a $W^\pm/Z^0$ boson.

2.3.4 Higgs boson decay

The probability (branching ratio) to which particles the Higgs boson is decaying according to the Standard Model (see Fig. 2.4) is strongly dependent on its mass. However, the mass is a priori not fixed in the SM and needs to be determined experimentally. In 2012 the ATLAS and CMS collaboration announced the discovery [16–18] of a boson



(a) Gluon-gluon fusion

(b) Vector-boson fusion

(c) $t\bar{t}$ fusion

(d) Higgs strahlung

Fig. 2.3: Feynman diagrams of the dominant SM Higgs production modes at the LHC.

with a mass of $m_H = 125\,\text{GeV}$. Spin and CP properties comply with those of the SM Higgs boson [19]. This leads to the branching ratios listed in Tab. 2.4. The most common decay for a Higgs boson with a mass of $m_H = 125\,\text{GeV}$ is into bottom-antibottom ($b\bar{b}$) quarks with a branching ratio of 0.577. The branching ratio of the decay to a tau-antitau ($\tau\tau$) pair, which is 0.0637, is about one magnitude smaller than the branching ratio for $b\bar{b}$. It is the second most common fermion decay mode. The discovery of the Higgs boson was announced after measuring an excess in the $H \to \gamma\gamma$ and $H \to ZZ$ channels. The decay $H \to \gamma\gamma$ has a very small branching ratio. However, it is very important since the energy of a photon can be measured very precisely yielding an accurate mass reconstruction of the decaying particle.

Decaymode	$b\bar{b}$	$\tau\tau$	$\gamma\gamma$	Zγ	WW	ZZ
Branching ratio	0.578	0.0637	0.0023	0.0016	0.216	0.0267

Tab. 2.4: Predicted branching ratios [20] for a Higgs boson with mass $m_H = 125\,\text{GeV}$.

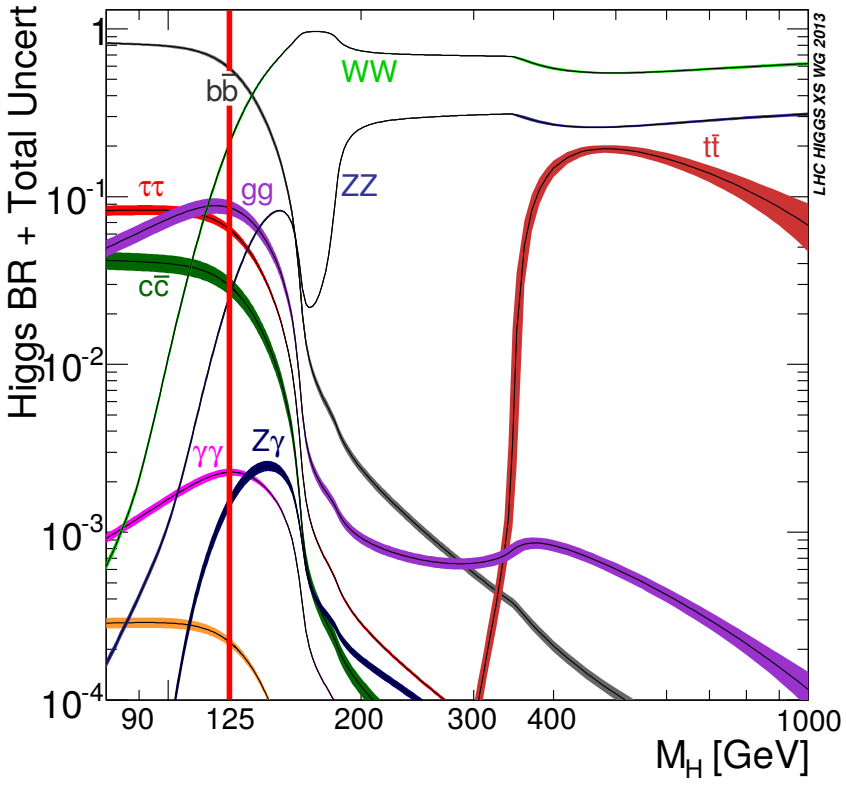


Fig. 2.4: Higgs boson decay branching ratios as a function of the Higgs boson mass m_H [21].

2.3.5 Evidence for a Higgs boson decaying into a pair of tau leptons

The fermion masses are generated via the Yukawa couplings [22] between the Higgs field and the fermionic fields in the SM. The measurements of these couplings is essential to identify the new boson as the SM Higgs boson. The most promising decay mode to measure one of these couplings is $H \to \tau\tau$. The ATLAS and CMS experiments found evidence [23, 24] for this decay. The experimental results as well as the analysis strategies are discussed in the following.

ATLAS search

The search at the ATLAS experiment [25] for the SM Higgs boson decaying to a pair of tau leptons was performed in the $H \to \tau\tau \to LL'$ final states (where L denotes an electron e, muon μ or hadronically decaying tau lepton τ_h) using the complete dataset collected at $\sqrt{s} = 8\,\mathrm{TeV}$ which corresponds to an integrated luminosity of $\mathcal{L} = 20.3\,\mathrm{fb}^{-1}$ [23]. After preselection and categorization of the $H \to \tau\tau$ events by production mode a *Boosted Decision Tree* (BDT) *Multivariate Analysis* (MVA) technique is used to extract the signal from the remaining background events [26–28]. For the analysis the signal strength parameter μ is used which is defined as the ratio of the measured cross section times the branching ratio normalized to the SM cross section times the branching ratio for $H \to \tau\tau$. A value of $\mu = 1$ corresponds to the presence of an SM Higgs boson signal and $\mu = 0$ to the absence of the Higgs boson. The final result is a signal strength parameter of $\mu = 1.43^{+0.31}_{-0.29}(\mathrm{stat.})^{+0.41}_{-0.30}(\mathrm{syst.})$ for a Higgs boson mass of $m_H = 125\,\mathrm{GeV}$. This corresponds to a deviation from the background-only hypothesis of $4.1\,\sigma$.

CMS search

The search for a Higgs boson decaying into a pair of tau leptons with the CMS experiment is, similar to the ATLAS experiment, performed in all six LL' final states ($e\tau_\mathrm{h}$, $\mu\tau_\mathrm{h}$, $\tau_\mathrm{h}\tau_\mathrm{h}$, $e\mu$, ee and $\mu\mu$) from a Higgs boson produced via VBF and ggF. Furthermore, the associated vector boson production mode (see Fig. 2.3d) is included by requiring additional leptons in the final state. These originate from a decaying W or Z boson. For the analysis the entire dataset collected in 2011 and 2012 is used which corresponds to an integrated luminosity of $\mathcal{L} = 4.9\,\mathrm{fb}^{-1}$ for 7 TeV and $\mathcal{L} = 19.7\,\mathrm{fb}^{-1}$ for 8 TeV [24]. To maximize sensitivity the events are classified according to a number of kinematic quantities and by the number of jets in the final state. The signal is extracted from the invariant mass distribution of the tau lepton pair obtained from the four momenta of L and L' as well as the missing transverse energy vector. The final result, as shown in Fig. 2.5, for a Higgs boson decaying into a pair of tau leptons is an excess of events compared to the background-only hypothesis corresponding to a significance of $3.2\,\sigma$ for a Higgs boson mass of $m_H = 125\,\mathrm{GeV}$.

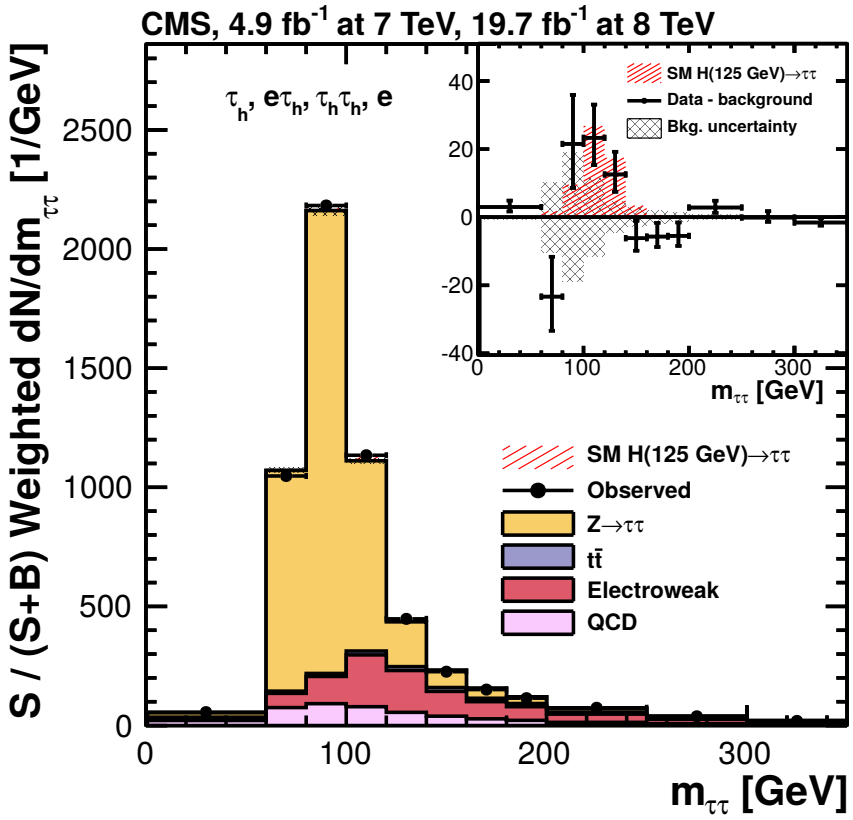


Fig. 2.5: Combined observed and predicted $m_{\tau\tau}$ distributions for the $\mu\tau_{\mathrm{h}}$, $e\tau_{\mathrm{h}}$, $\tau_{\mathrm{h}}\tau_{\mathrm{h}}$ and $e\mu$ channels [24].

3 Experimental setup at CERN

3.1 Large Hadron Collider

The accelerator complex at CERN is at the moment the most powerful particle collider in the world. Its main goal is to test predictions, made by theories of particle and high-energy physics like electroweak symmetry breaking, supersymmetry or dark matter. Since the establishment of CERN in 1954, several groundbreaking achievements were made.

- 1983: Discovery of the W and Z bosons [29].
- 1995: Creation of antihydrogen atoms for the first time [30].
- 1999: Discovery of direct CP violation [14].
- 2012: Discovery of the Higgs boson [16–18].

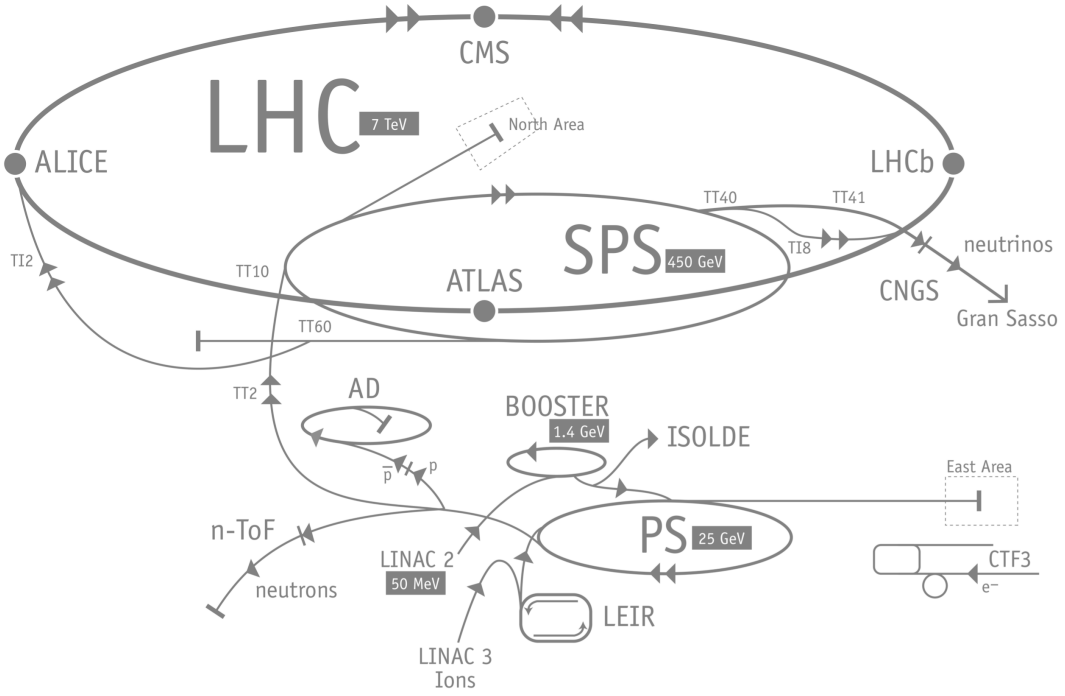


Fig. 3.1: Map of the CERN accelerator complex [31]. The acronyms of the individual particle accelerators are described in the text.

The LHC, which is short for Large Hadron Collider, is only the last element of a chain of particle accelerators as one can see in Figure 3.1. Each element accelerates

the particles to a higher energy until they reach their maximum energy in the LHC itself. The individual accelerators are:

- LINAC 2, Linear Accelerator, 50 MeV.
- PSB, Proton Synchrotron Booster, 1.4 GeV.
- PS, Proton Synchrotron, 25 GeV.
- SPS, Super Proton Synchrotron, 450 GeV.
- LHC, Large Hadron Collider, up to 13 TeV.

The first four acceleration processes take about 20 min. Afterwards, it takes another 20 min to reach the maximum beam energy. The maximum center-of-mass energy was increased step by step from 7 TeV in 2010 and 2011 to 8 TeV in 2012 and to 13 TeV in 2015 after the first long shutdown. At full operating power the LHC is designed to reach a center of mass energy of 14 TeV. In Tab. 3.1 the complete timetable including the corresponding luminosities is shown. Inside the 27 km long accelerator ring, two beams consisting of either protons or heavy ions travel in opposite direction. The beams consist of small proton bunches that cross at four different interaction points every 25 ns. This is where the four detectors are located.

- ALICE, A Large Ion Collider Experiment.
- LHCb, LHC beauty.
- ATLAS, A Toroidal LHC Apparatus.
- CMS, Compact Muon Solenoid.

Year	Energy [TeV]	Pile-Up[1]	Instantaneous Luminosity $[cm^{-2}s^{-1}]$
2011	7	7	$0.3 \cdot 10^{34}$
2012	8	21	$0.7 \cdot 10^{34}$
$2015 - 18$	$13 - 14$	43	$1.6 \cdot 10^{34}$
$2020 - 22$	14	$50 - 80$	$2 - 3 \cdot 10^{34}$
$2025 - 28$	14	$140 - 200$	$5 - 7 \cdot 10^{34}$

Tab. 3.1: Past and future development of the center-of-mass energy at the LHC including the corresponding luminosity.

3.2 The CMS detector

The CMS experiment, like ATLAS, is one of two multipurpose particle physics detectors. The detector requirements can be summarized as follows [32, 33]:

[1] Average number of concurrent proton-proton interactions in a single beam crossing.

- A good muon identification and momentum resolution over a wide range of transverse momenta and a good dimuon mass resolution of at least 1% at 100 GeV is required.

- The *inner tracking system* needs a good charged particle momentum resolution and reconstruction efficiency. For efficient high-level triggering and offline identification of τ leptons and b-jets, pixel detectors close to the interaction region are required.

- The *Electromagnetic Calorimeter* (ECAL) needs a good energy resolution and a good diphoton and dielectron mass resolution of around 1% at 100 GeV. Furthermore, a wide geometric coverage of $|\eta| < 3$ is needed.

- To get a good E_T^{miss} (transverse missing energy) and dijet resolution, the *Hadronic Calorimeter* (HCAL) needs a large geometric coverage of $|\eta| < 5$ and a fine lateral resolution of $\Delta\eta < 0.1$ and $\Delta\phi < 0.1$ (where ϕ is the azimuthal angle)

3.3 Detector components

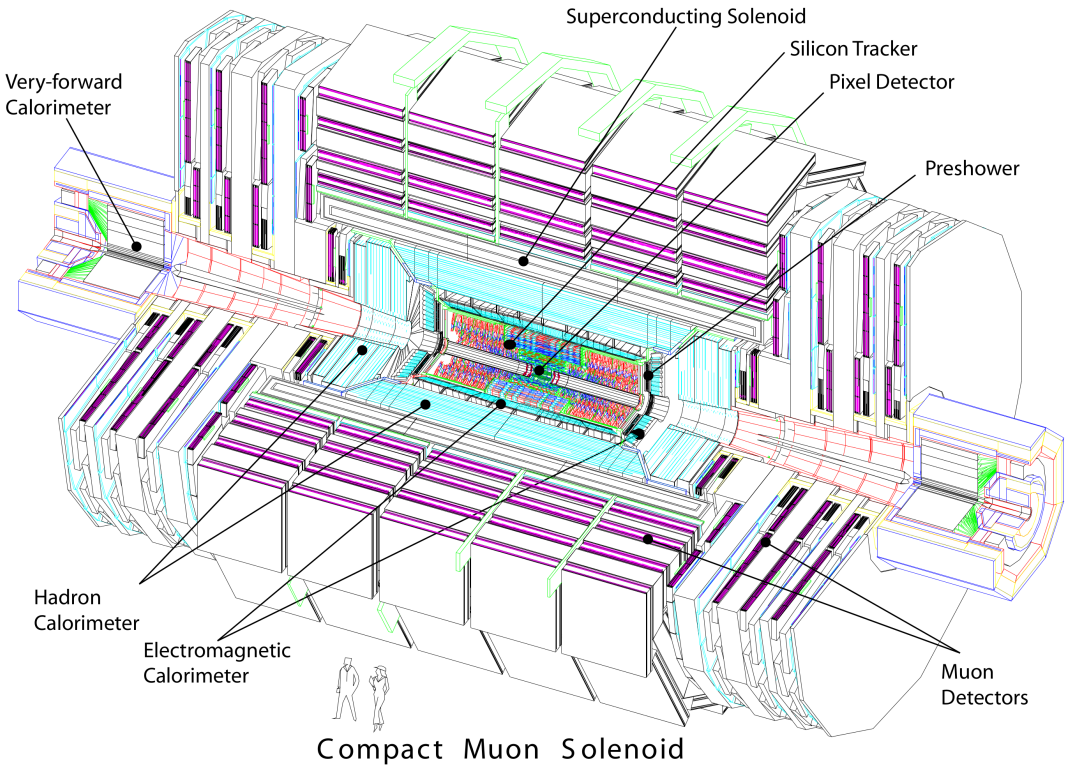


Fig. 3.2: Overview of the CMS detector [32].

3.3.1 Detector overview

As one can see in Fig. 3.2, the CMS detector shows the typical onion-like structure of most particle detectors. The main features of CMS are a silicon-based inner tracking system, the scintillating-crystal electromagnetic calorimeter and the hadron calorimeter, a high magnetic field solenoid as well as the muon system. In the Secs. 3.3.2 – 3.3.6 an overview of the different subdetectors is given. Figure 3.3 shows different particle types propagating through the CMS detector and interacting with the individual detector parts.

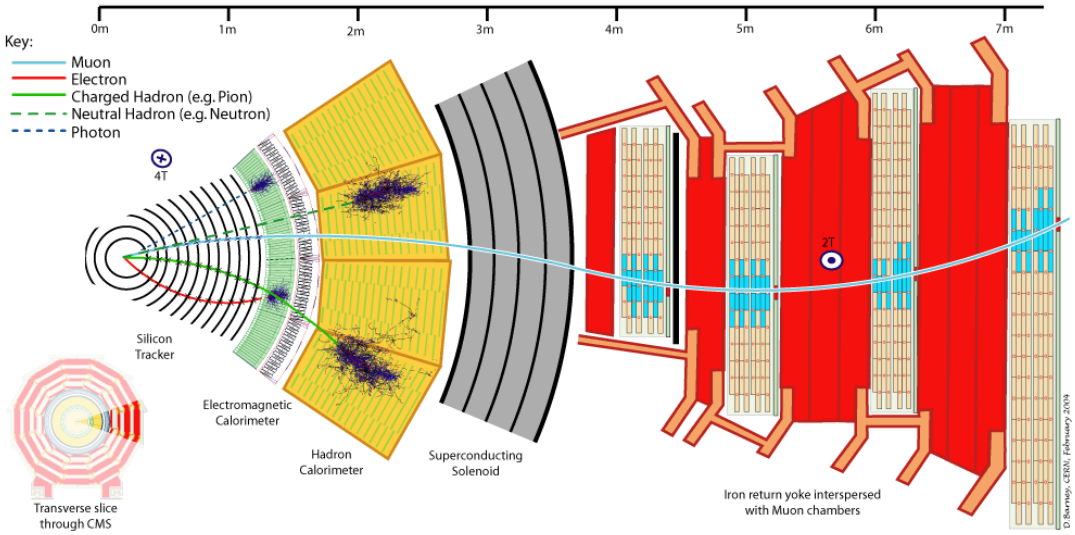


Fig. 3.3: Tracks through the CMS detector of different particles [34].

3.3.2 Inner Tracking System

The innermost subdetector of CMS is the tracking system. It is responsible for measuring the trajectories of charged particles as well as reconstruction of secondary vertices. To satisfy the requirements on granularity, speed and radiation hardness, coming with the intense particle flux, the inner tracking system is solely based on silicon detector technology. Since the magnetic field created from the superconducting solenoid can be considered homogeneous, the momenta of these particles can be obtained directly from the curvature of their trajectories. To obtain the trajectory, the position of the particle is measured several times. The flux of the particles decreases with $1/r^2$, therefore, the highest precision is needed close to the beam pipe. The innermost part of the inner tracker consist of three barrel pixel detectors (see Fig. 3.4) located at radii of 4.4 cm, 7.3 cm and 10.2 cm. The pixel detectors are surrounded by the *Tracker Inner Barrel and Disks* (TIB/TID) and the *Tracker Outer Barrel* (TOB), as illustrated in Fig. 3.5, extending the outward radius to 110 cm. The pixel detector as well as the TIB and TOB are completed by the endcaps (TEC+ and TEC-) extending the acceptance up to a pseudorapidity of $\eta < 2.5$ [32, 33].

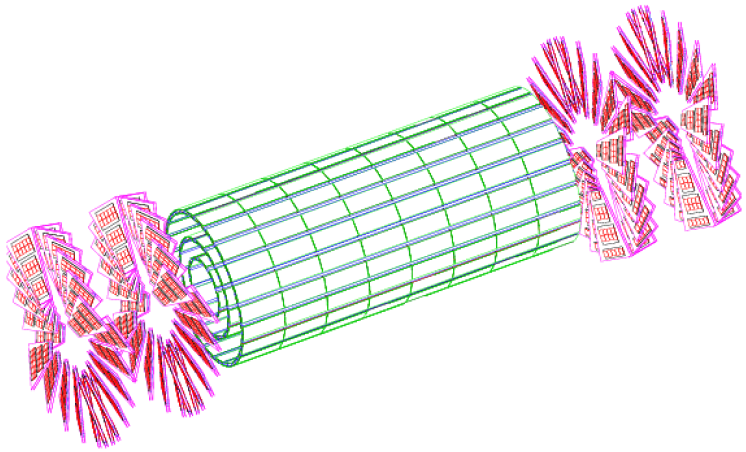


Fig. 3.4: Layout of the barrel pixel detector extended by endcaps [32].

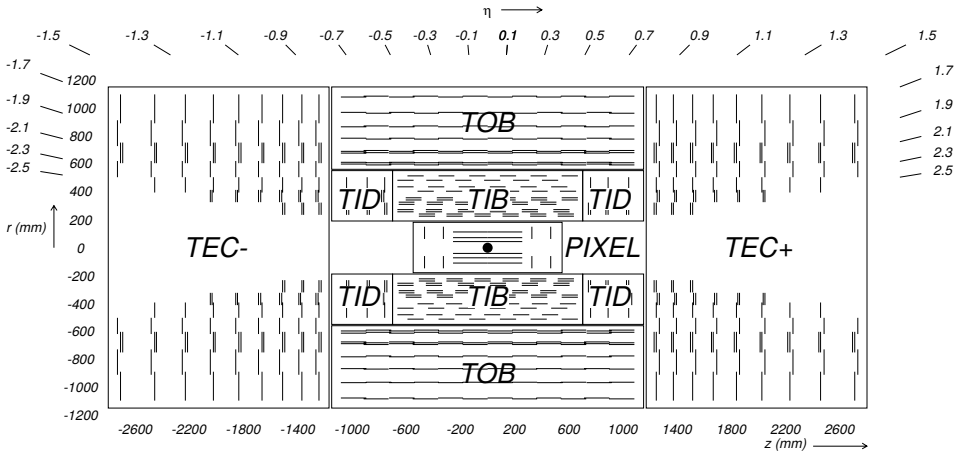


Fig. 3.5: Schematic of the inner tracker [33].

3.3.3 Electromagnetic Calorimeter (ECAL)

The ECAL is the next layer after the inner tracking system. It is designed to measure electrons and photons as well as charged hadrons. For this purpose it uses scintillating lead-tungstate crystals. They emit 80% of the scintillation light in 25 ns which is in the same order of magnitude as the bunch crossing time. This light is detected by *Avalanche Photodiodes* (APD) in the barrel region (EB), which covers the region $|\eta| < 1.479$; *Vacuum Phototriodes* (VPT) are installed to cover the rapidity range from $1.479 < |\eta| < 3$ in the endcap (EE). In front of the endcap, a preshower system is installed for π^0 rejection. In Fig.3.6 the transverse section of the ECAL is shown [32, 33].

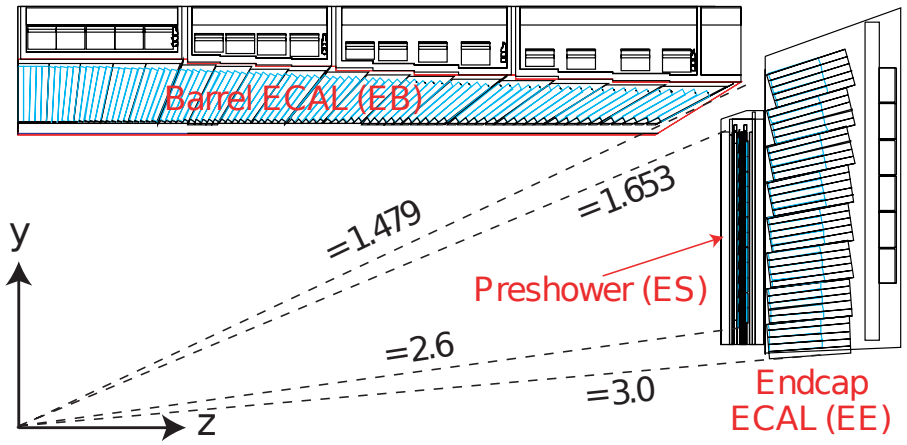


Fig. 3.6: Transverse section of the electromagnetic calorimeter [32].

3.3.4 Hadronic Calorimeter (HCAL)

The HCAL is important for the measurement of hadron jets as well as missing energies associated to neutrinos or other not detectable particles. The HCAL is split into four subsystems which are the *hadronic barrel* (HB), the *hadronic endcap* (HE), *hadronic forward* (HF) and the *hadronic outer barrel* (HO) calorimeter. The HB is located between the ECAL and the superconducting solenoid and covers a range up to $|\eta| < 1.3$. It consists of 16 brass alloy absorber layers interspersed with plastic scintillator tiles which are read out by embedded wavelength-shifting fibers. Brass was chosen as absorber material because of its short hadron interaction length. The HE extends the coverage up to $|\eta| < 3$ and is similarly segmented and read out as the HB. The HF is especially designed to measured the energetic forward flux and covers a range $3 < |\eta| < 5$. The HO is located outside the solenoid and its purpose is to compensate the lack of absorber material in the HB [32, 33].

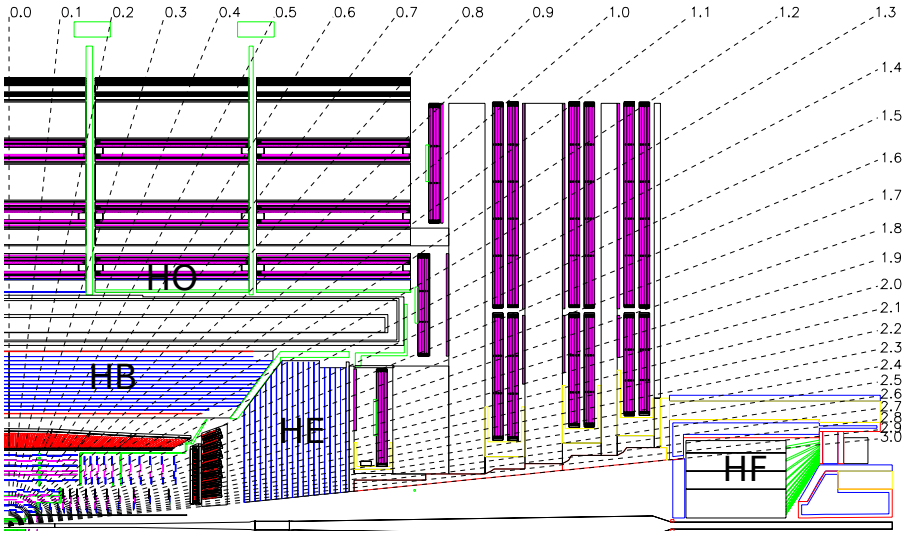


Fig. 3.7: Transverse section of the hadronic calorimeter [33].

3.3.5 Magnet

In order to unambiguously determine the sign of charged particles, a momentum resolution $\Delta p/p$ of approximatly 10% at 1 TeV is needed. This can be achieved with a large bending power, i.e. a large magnetic field. For this task, a superconducting solenoid, specifically a high-purity aluminium-stabilised conductor with indirect cooling, was chosen. The conductor is placed around the calorimeter and is surrounded by a 10 t steel yoke which closes the magnetic field lines. The weight of the steel yoke is the main contribution to the overall weight of the CMS detector. In Tab. 3.2 the main parameters of the solenoid are summarized [32, 33].

Magnetic flux density	3.8 T
Inner bore	5.9 m
Length	12.9 m
Number of turns	2168
Current	19.5 kA
Stored energy	2.7 GJ

Tab. 3.2: Summary of the main parameters of the solenoid [32]

3.3.6 Muon System

Since muons are an essential part of nearly all signatures from physical signals interesting for the CMS experiment, it is crucial to determine muons and their momentum unambiguously. The layout of the muon system is given in Fig. 3.8. In the muon system, three types of gaseous detectors are used. The detector types are chosen because of the large surface to be covered and the different radiation regions. In the barrel region ($|\eta| < 1.2$), Drift Tubes (DTs) are installed. In the endcap region ($0.9 \leq |\eta| \leq 2.4$) the DTs are superseded by Cathode Strip Chambers (CSCs). Additional to these two detectors, in both the barrel and in the endcap region, Resistive Plate Chambers (RPCs) are installed for $|\eta| < 1.6$ [32, 33].

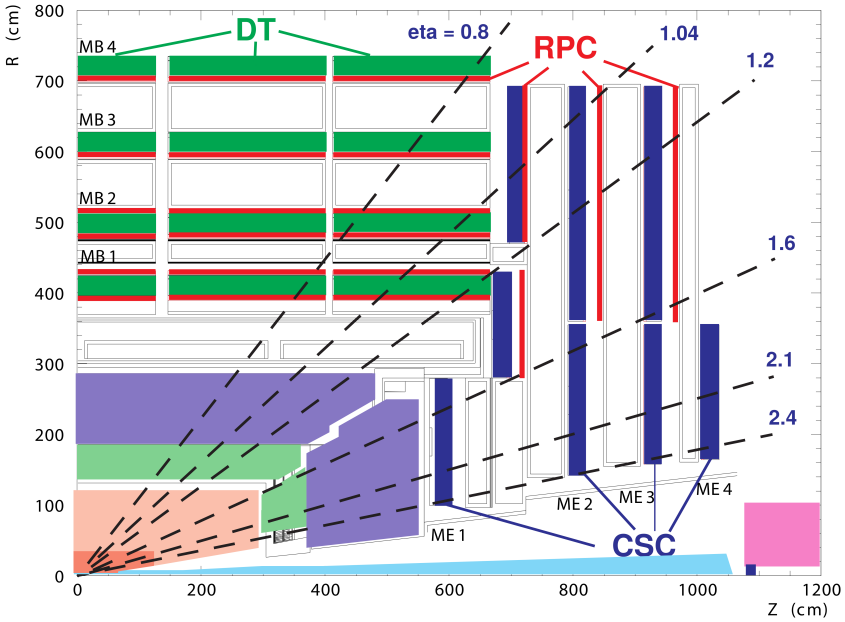


Fig. 3.8: Layout of one quarter of the CMS muon system [32].

4 Artificial neural networks

4.1 The artificial neuron

The artificial neuron is the basic building block of the neural network used in this thesis. The neuron is a computational unit which calculates the output or activation $h(x)$ based on the values of units it is connected to. For a single neuron, as shown in Fig. 4.1, the input is given by the vector x_i. The connections between the input variables x_i and the neuron are called weights w_i. The actual value used for the calculation of the activation $h(x)$ is the sum of the products of the input variable x_i and the corresponding weight w_i. This product is called pre-activation $p(x)$. The pre-activation can be written as

$$p(x) = b + \sum_{i=1}^{N} w_i x_i, \tag{4.1}$$

where b is called neuron bias. The bias can be used to set a specific offset on the pre-activation and is usually a constant. The neuron uses the value of the pre-activation $p(\mathbf{x})$ and applies the so called activation function a to get the actual activation $h(\mathbf{x})$ of the neuron:

$$h(\mathbf{x}) = a(p(\mathbf{x})) = a\left(b + \sum_{i=1}^{N} w_i x_i\right) \tag{4.2}$$

In Sec. 4.2 a short overview of the most popular activation functions is given.

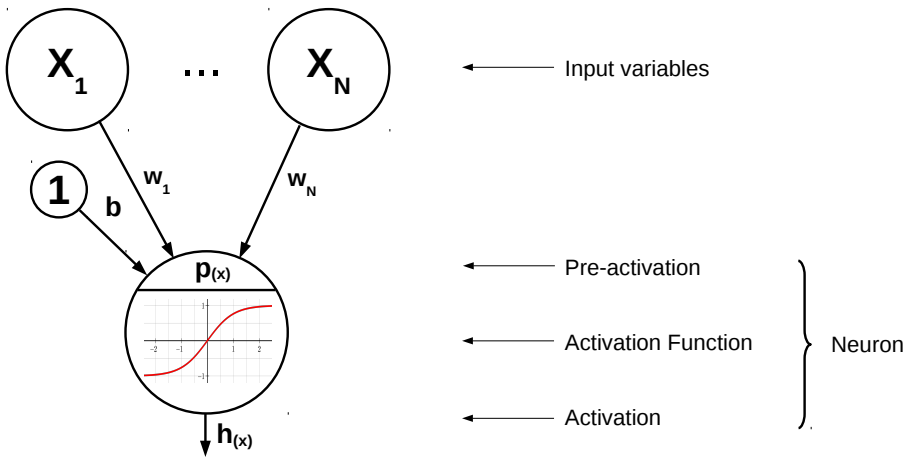


Fig. 4.1: Components of a single neuron.

4.2 Activation functions

4.2.1 Linear function

The linear or identity function (see Fig. 4.2a) simply takes the value of the pre-activation p and reproduces it. Since it does not introduce any non-linearities in the computation it is rarely used in a neural network. However, it is often used as the neuron of the input variables:

$$a(p) = p. \tag{4.3}$$

The linear activation function does not have any lower or upper boundaries.

4.2.2 Rectified linear function

The rectified linear function (see Fig. 4.2b) as described by Glorot and Bengio [35] introduces sparsity in the network architecture. For positive values of the pre-activation p it is simply the linear function. For negative values it is always 0 which leads to the above mentioned sparsity. This means, that for negative values of the pre-activation the connections from such a neuron to other neurons are omitted:

$$a(p) = \begin{cases} p & \text{if } p \geq 0 \\ 0 & \text{else} \end{cases}. \tag{4.4}$$

4.2.3 Sigmoid

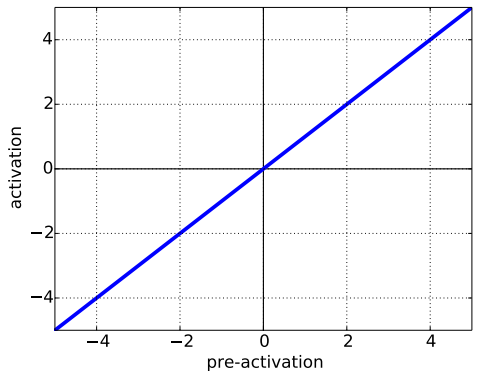
The sigmoid activation function (see Fig. 4.2c) takes the value of the pre-activation p and transforms it according to eq. 4.5. The activation $a(p)$ can take values between the lower and upper bound of 0 and 1. Therefore, it is always positive and strictly monotonously increasing:

$$a(p) = \frac{1}{1 + e^{-p}}. \tag{4.5}$$

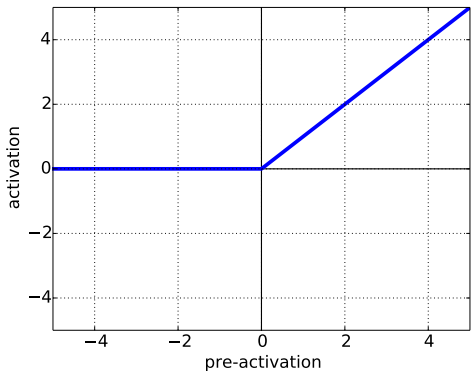
4.2.4 Hyperbolic tangent

The Hyperbolic tangent (Tanh) (see Fig. 4.2d) is similar to the sigmoid function but the activation a can take values between the lower and upper bound of -1 and 1. It is also strictly monotonously increasing but can be positive or negative:

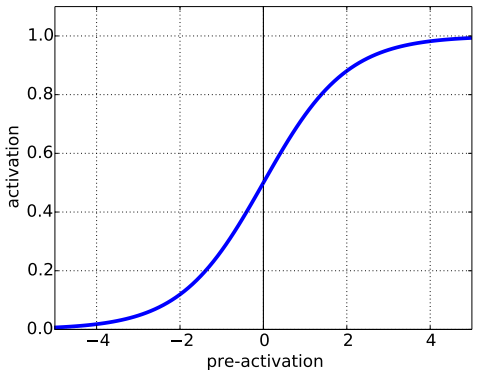
$$a(p) = \frac{e^{2p} - 1}{e^{2p} + 1}. \tag{4.6}$$



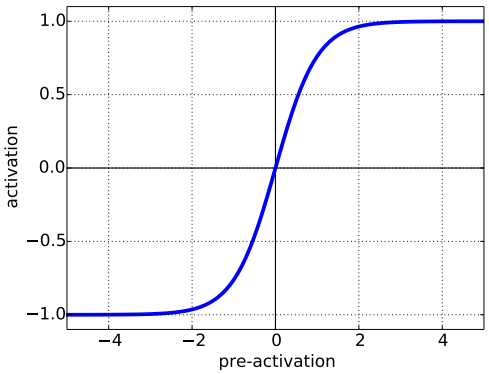
(a) Linear activation function



(b) Rectified linear activation function



(c) Sigmoid activation function



(d) Hyperbolic tangent activation function

Fig. 4.2: Illustration of commonly used activation functions.

4.3 Multilayer perceptron

4.3.1 The perceptron

Prior to introducing the multilayer perceptron, it is necessary to discuss the capacity of a single neuron and its application in a perceptron [36]. In fact, one of the simplest perceptrons is a single neuron as shown in Fig. 4.1 with $N = 2$ input variables. If x_1 and x_2 are binary inputs this perceptron can be used for binary classification. An example for such a binary classification is given in Fig. 4.3 with the logic AND function. As one can see it is possible to separate the output in two classes by drawing the bold dashed line. The first problem occurs when trying to represent the logic XOR with

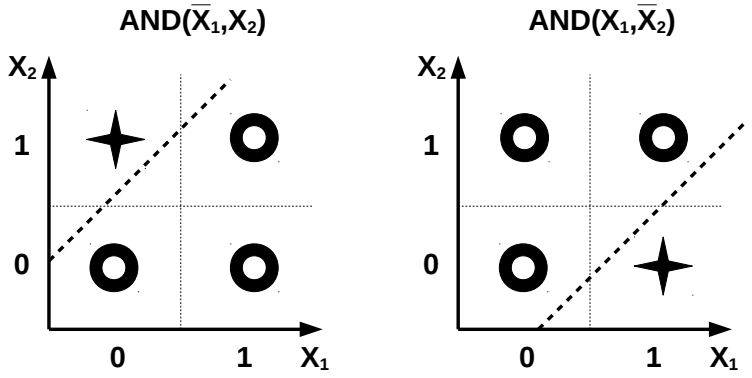


Fig. 4.3: Karnaugh diagrams of an AND function where one of the two inputs x_1, x_2 is negated.

such a simple perceptron. As shown in Fig. 4.4, it is not possible to separate the two classes by a single straight line. Thus, a single neuron is not capable of representing the XOR function. The solution is to transform the problem into a better representation yielding a combination of two AND functions in this case, as shown in Fig. 4.4. What happens is that the values for $x_1 = x_2 = 0$ and $x_1 = x_2 = 1$ collapse to a single point and the problem again becomes linearly separable. The actual transformation

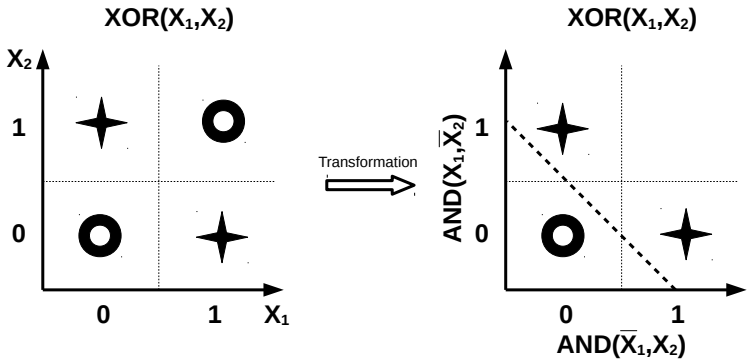


Fig. 4.4: Karnaugh diagram of the XOR function with the inputs x_1, x_2 (left) and of the XOR function with two AND functions as input (right).

performed to solve this problem was the insertion of an additional layer (see Fig. 4.5)

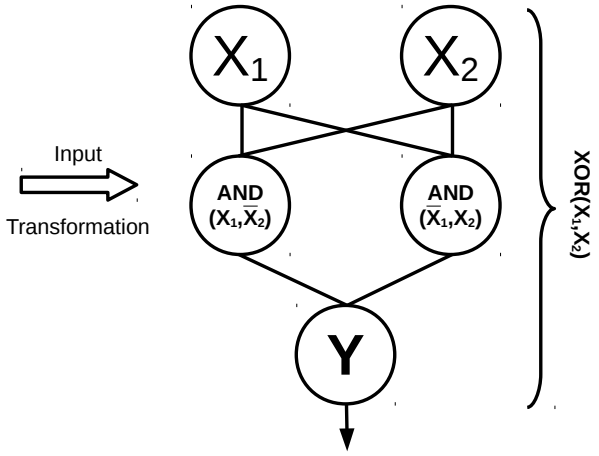


Fig. 4.5: Neural network with one hidden layer which is able to represent the XOR function by combining two AND functions.

between input and output. This leads to the definition of the multilayer perceptron.

4.3.2 The multilayer perceptron

As seen in the previous section, it is possible to solve more complex problems by introducing a so-called hidden layer. The basic idea is that this hidden layer performs a transformation from a complex problem to a simpler one. The number of neurons inside the hidden layer (they will be referred to as *nodes* from this point on) depends on the complexity of the problem. The multilayer perceptron is a so called feed-forward neural network. In a feed-forward network the output of one layer is only connected to the input of the next deeper layer. Additionally no loop connections and no connections in the same layer are allowed. In Fig. 4.6 an example of a multilayer perceptron is shown where the input layer is only connected to the first hidden layer and the hidden layer is only connected to the output layer. The connections are again described by weights. In a more formal way, analogous to Russel and Norvig [37], the transformation to the

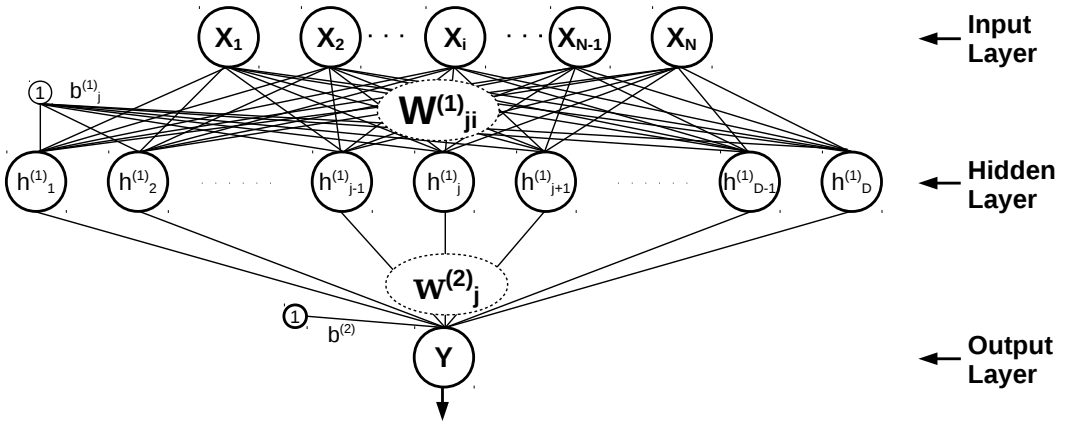


Fig. 4.6: General form of a perceptron with one hidden layer.

new representation can now be written as

$$h_j^{(1)}(\mathbf{x}) = a_j^{(1)}\left(p_j^{(1)}(\mathbf{x})\right) = a_j^{(1)}\left(b_j^{(1)} + \sum_{i=1}^{N} W_{ji}^{(1)} x_i\right), \tag{4.7}$$

where the output activation of the node $h_j^{(1)}(\mathbf{x})$ is calculated similarly to eq. 4.2 by applying the activation function $a_j^{(1)}$ on the pre-activation $p_j^{(1)}(\mathbf{x})$. The activation function $a_j^{(1)}$ can be any of the functions introduced in Sec. 4.2 and may be different for each single node. However, usually one specific activation function is chosen for all hidden layers reducing $a_j^{(1)}$ to a. Furthermore, it is possible that each node has a different bias $b_j^{(1)}$. Nevertheless, in most multilayer perceptrons the bias is completely omitted in the whole network. The weights connecting two layers are described by the matrix $W_{ji}^{(1)}$. The last step is to calculate the activation $y(x)$ of the complete network

$$y(x) = a^{(2)}\left(b^{(2)} + \sum_{j=1}^{D} w_j^{(2)} h_j^{(1)}(x)\right), \tag{4.8}$$

where $a^{(2)}$ is the activation function of the output layer and $h_j^{(1)}(\mathbf{x})$ is the activation of the hidden layer. For the sake of completeness, the bias was also included in eq. 4.8. Since the output is only a single node, the weights reduce again from matrix to vector form $w_j^{(2)}$. This equations allows to perform a forward propagation meaning that the calculation of the activation $h_j^{(1)}(\mathbf{x})$ of all nodes as well as the activation of the output layer $y(\mathbf{x})$ is possible. Thus the concept of a perceptron with one hidden layer can now be generalized to a multilayer perceptron.

Assuming a multilayer perceptron with L hidden layers an index k is introduced which takes values from $1 \dots L$. The numbers $N^{(k)}$ represent the number of nodes in the specific layer k. The output activation of the input layer is defined as

$$h_i^{(0)} = x_i. \tag{4.9}$$

The output activation of all hidden layers can now be written as:

$$h_j^{(k)}(\mathbf{x}) = a_j^{(k)}\left(b_j^{(k)} + \sum_{j=1}^{N^{(k)}} \sum_{i=1}^{N^{(k-1)}} W_{ji}^{(k)} h_i^{(k-1)}(\mathbf{x})\right). \tag{4.10}$$

The final step is to calculate the activation $y(x)$ of the complete network:

$$y(\mathbf{x}) = h^{(L+1)}(\mathbf{x}) = a^{(L+1)}\left(b^{(L+1)} + \sum_{i=1}^{N^{(L)}} w_i^{(L+1)} h_i^{(L)}(\mathbf{x})\right). \tag{4.11}$$

The activation function of the output layer is frequently different from the activation functions in the hidden layers. Therefore, it is convenient to write the activation function of the output layer $a^{(L+1)} = o$ and all other activation functions $a_j^{(k)} = a$.

With this assumption, and also omitting all biases, one gets a very compact form of the multilayer perceptron forward propagation:

$$
\begin{aligned}
h_i^{(0)} &= x_i, \\
h_j^{(k)}(\mathbf{x}) &= a\left(\sum_{j=1}^{N^{(k)}} \sum_{i=1}^{N^{(k-1)}} W_{ji}^{(k)} h_i^{(k-1)}(\mathbf{x})\right), \\
y(\mathbf{x}) &= o\left(\sum_{i=1}^{N^{(L)}} w_i^{(L+1)} h_i^{(L)}(\mathbf{x})\right).
\end{aligned}
\tag{4.12}
$$

4.3.3 Supervised learning

Consider a given training set which consists of N example input-output pairs $\left(\mathbf{x}^{(j)}, y^{(j)}\right)$. The output $y^{(j)}$ was generated by an unknown function $F\left(\mathbf{x}^{(j)}\right) = y^{(j)}$ and is called target output or label. The task of supervised learning is to discover a function H which approximates the unknown function F. The function H is called hypothesis and corresponds to a configuration of weights and biases of a multilayer perceptron. In fact, multilayer perceptrons are universal appriximators in the sense that any non-pathologic (measurable) function F can be approximated to arbitrary precision by a hypothesis H, as long as the network has at least one hidden layer with a sufficient number of hidden neurons and a bounded activation function. The theorem in its original form [38] does not tell us how many neurons are actually required to reach a certain precision or what a good activation could look like. Still, it is a powerful argument for using neural networks to approximate functions F for which no closed form is available. The search through the space of possible hypotheses for one that only performs well not on the training set but also on examples beyond this set is called learning or training. To measure the accuracy of a hypothesis a test set is used which is distinct from the training set. To this end a cost function [39]

$$
C(\theta) = \frac{1}{n} \sum_{j=1}^{n} \mathcal{L}\left(\hat{y}_\theta^{(j)}, y^{(j)}\right) \tag{4.13}
$$

is introduced, where the parameter $\theta = \left(W^{(1)}, b^{(1)}, \ldots, W^{(L+1)}, b^{(L+1)}\right)$ represents the weights and biases of the neural network and is therefore the representation of all interconnections inside the network. The functional $\mathcal{L}$ is called loss functional. The parameter $\hat{y}_\theta^{(j)}$ is computed from the input vector $\mathbf{x}^{(j)}$ using the configuration θ by performing a forward propagation. The parameter $\hat{y}_\theta^{(j)}$ is identical to the activation of the output layer in a multilayer perceptron. The value $y^{(j)}$ represents the target value of the output layer one would expect from a forward propagation of $\mathbf{x}^{(j)}$ by an ideal multi-layer perceptron that reproduces the function F exactly. The loss functional compares the output $\hat{y}_\theta^{(j)}$ after a forward propagation with the target output $y^{(j)}$ and provides information on how well these two values agree with each other. The cost function is the average over the loss functionals of all $\left(\hat{y}_\theta^{(j)}, y^{(j)}\right)$ pairs and is therefore

a measure of the accuracy of the hypothesis. The optimum of the neural network configuration is obtained by finding the minimum of the cost function with respect to the parameter θ:

$$\frac{\partial C(\theta)}{\partial \theta} = 0. \tag{4.14}$$

Training of the neural network by finding the minimum of the cost function is equal to finding the best hypothesis for the given problem. However, if the neural network is trained in such a way that the training set is approximated to carefully a problem called overtraining occurs which will be discussed in Sec. 5.4.3.

4.3.4 Stochastic gradient descent

In general it is not possible to find the solution to the cost function minimization problem analytically. Therefore, numerical optimization methods are needed. The simplest one is called *gradient descent*, represented by the equation

$$\theta^{l+1} = \theta^l - \epsilon_l \frac{\partial C(\theta^l)}{\partial \theta^l}, \tag{4.15}$$

where the parameter θ^l is the configuration θ at the iteration step l. The factor ϵ_l is a scalar called learning rate. It determines what fraction of the gradient in either eq. 4.15 or eq. 4.16 is used to update the parameter θ. The learning rate can either be chosen during training as constant or to decrease over time. If the learning rate is chosen too large, the iterations will lead away from the minimum. But if the learning rate is chosen too small, training will be slow since it takes long to reach convergence. Since the gradient is always directed to the steepest descent, this method guarantees the finding of a local minimum as long as the learning rate is chosen well. A fast and stable variation of the gradient descent is the *Stochastic Gradient Descent* (SGD):

$$\theta^{l+1} = \theta^l - \epsilon_l \frac{\partial \mathcal{L}\left(\hat{y}_{\theta^l}, y\right)}{\partial \theta^l}. \tag{4.16}$$

This method exploits the fact that the cost function is the average of the loss functionals. The SGD is calculated after each example. The SGD algorithm is typically much faster then the gradient descent since it is calculated after each example and therefore updates θ more frequently. Whereas the update direction of the SGD is fluctuating more than for the gradient descent the expectation value of these updates yields the same update as for the later. A good tradeoff between gradient descent and stochastic gradient descent is the *minibatch stochastic gradient descent*. Instead of calculating the gradient after each example, it evaluates the gradient of a small subset (minibatch) from the complete set of examples. This yields a less noisy gradient, but due to the high number of updates convergence is obtained more quickly than for gradient descent.

Kullback-Leibler divergence

The loss functional used in this thesis is the Kullback-Leibler divergence [40] which measures the difference between two probability distributions. The Kullback-Leibler

divergence $KL(P||Q)$, where P is defined by the estimated values $\hat{y}_\theta$ and Q is defined by the target values y, is the mean of divergences $KL(\hat{y}_\theta||y)$ calculated for each example:

$$KL(P||Q) = \frac{1}{N} \sum KL(\hat{y}_\theta||y), \tag{4.17}$$

$$KL(\hat{y}_\theta||y) = \hat{y}_\theta \log \hat{y}_\theta - \hat{y}_\theta \log y + (1 - \hat{y}_\theta) \log(1 - \hat{y}_\theta) - (1 - \hat{y}_\theta) \log(1 - y). \tag{4.18}$$

Momentum

Momentum, as introduced by Hinton [41], is useful when the training algorithm is only slightly improving because it reaches a plateau or when the training algorithm is stuck in a local minimum. One can think of the training process as a heavy ball rolling down a hill (see Fig. 4.7). The momentum term αv^l adds an additional value based on previous updates. The parameter α is used to control the amount of momentum added to the update of θ. Similar to the learning rate, the momentum can be either chosen constant or as increasing over time:

$$v^{l+1} = \alpha v^l - \epsilon_l \frac{\partial L\left(\hat{y}_{\theta^l}, y\right)}{\partial \theta^l} \tag{4.19}$$

$$\theta^{l+1} = \theta^l + v^{l+1} \tag{4.20}$$

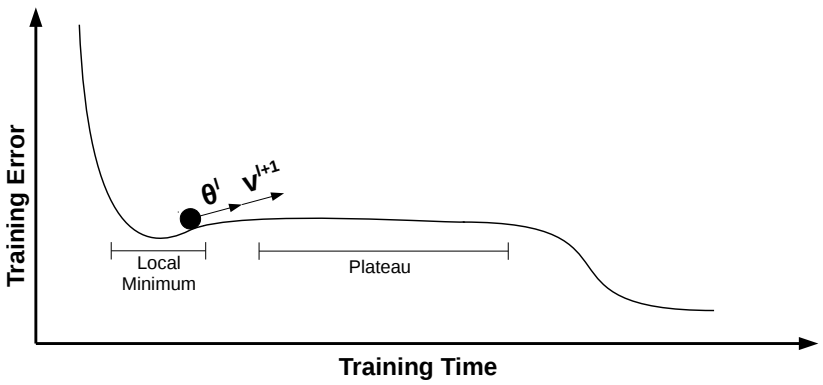


Fig. 4.7: Adding a momentum during a plateau helps reaching the minimum faster

4.3.5 Backpropagation of error

Up to now the algorithm to perform a forward propagation on a multilayer perceptron has been derived. The stochastic gradient descent is a procedure to optimize the performance of such a neural network. Thus, the only thing left is an effective algorithm to update the parameter θ which has already been introduced in Sec. 4.3.4 as the representation of all interconnections inside the neural network. Updating θ requires an update of all weights and biases. The underlaying algorithm performing this task

is called *backpropagation of error* [37]. Backpropagation is strictly speaking an inverse forward propagation with the main difference that during propagation from the L^{th} to the $(L-1)^{th}$ hidden layer the weights and biases are updated (see Fig. 4.8).

- At first a forward propagation is performed yielding all activations of all nodes in the neural network.
- The next step is to calculate the loss functional $L(\hat{y}_\theta, y)$ at the output layer and the error which is the gradient of the loss functional.
- This error is used to update the weights and biases between the output layer and the L^{th} hidden layer yielding a new error of the output of the L^{th} hidden layer.
- From this new error the gradient is computed which is used to update the weights and biases between the L^{th} hidden layer and the $(L-1)^{th}$ hidden layer.
- This procedure is repeated until all weights and biases are updated.

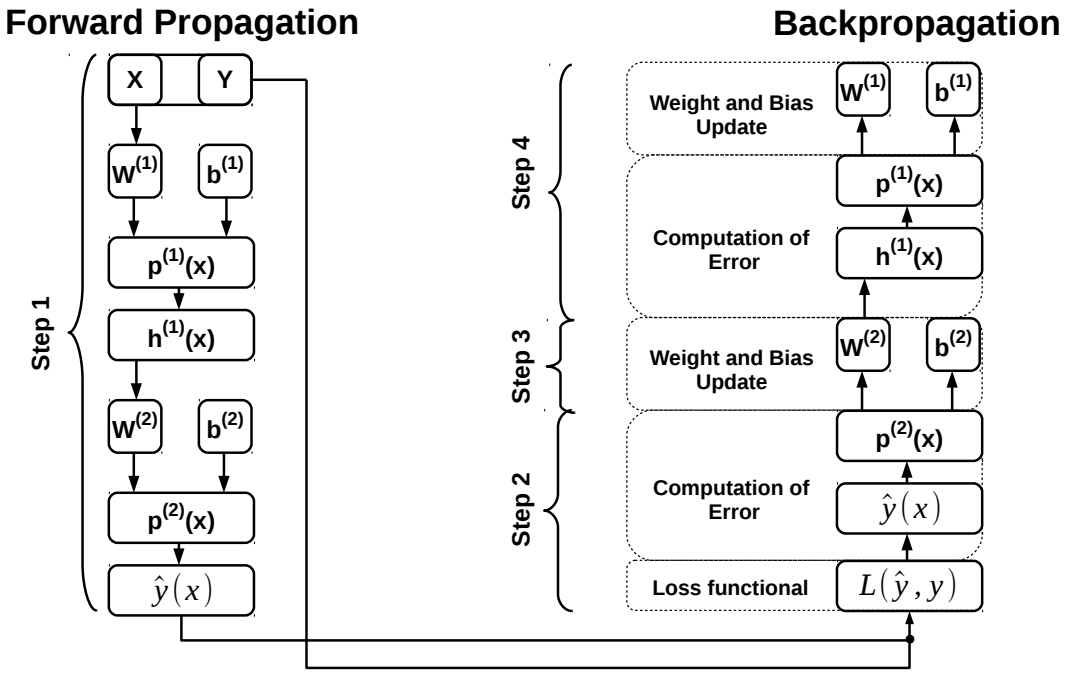


Fig. 4.8: Training of a neural network by applying forward and backpropagation

4.4 Deep learning

The concept of deep learning includes, amongst others, the research of neural networks with many hidden layers. Thus, all multilayer neural networks (including the multilayer perceptron) can be summarized as *Deep Neural Network* (DNN). In this thesis,

a neural network with one hidden layer will be referred to as shallow network, neural networks with more than one hidden layer as Deep Neural Networks. The idea of deep learning is derived from the way the human brain extracts complex information from e.g. visual or audible input. For example, the first information extracted from a picture could be low-level features such as lines and edges. This features are then transformed to geometric forms and afterwards to complex high-level features like faces and objects. After each step the information gets more complex until the complete picture is obtained. In a deep neural network something similar happens during training. Each additional hidden layer represents a more complex combination of features e.g.:

- **1st hidden layer** → lines, edges
- **2nd hidden layer** → geometric forms
- **nth hidden layer** → faces, objects
- **output layer** → description of the picture

In the late 1980s [42] first attempts to train a Deep Neural Network (DNN) with backpropagation were made, but without great success. The main problem was the vanishing gradient problem [43] arising from the fact that the backpropagated error decays exponentially from layer to layer yielding smaller updates every time. To compensate the vanishing gradient problem it is necessary to train sufficiently long. Thus, one needs considerable computation power to shorten training time. Insufficient computation power in the late 1980s was the bottleneck and the reason why research on DNNs was dropped back then. With the drastically increasing chip performance of CPUs and *Graphics Processing Units* (GPUs) as well as the availability of large training sets, deep learning has become feasible.

In high-energy physics a possible dataset of low-level features contains momenta, angles and energies of particles that are visible for the detector in use. However, this low-level data does not provide enough information to extract the desired signal events directly. Therefore, particles that are invisible for the detector (such as neutrinos) but crucial for the classification of an event need to be reconstructed by using the information provided by the low-level data. An example of a high-level feature is the reconstructed invariant mass of an unstable particle. By combining such high-level features it is possible to separate the desired events (signal events) from all other events (background events). It was shown [44] that a DNN is capable of extracting such high-level features directly from a dataset of low-level features, making it unnecessary to calculate them in the first hand. In summary the following scheme for a DNN in high-energy physics emerges:

- **1st hidden layer** → momenta, angles, energies,...
- **nth hidden layer** → missing energy, invariant mass,...
- **output layer** → classification signal/background

5 Multivariate analyses with deep neural networks

The purpose of this chapter is to summarize all relevant informations necessary to perform multivariate analysis with a deep neural network. First, the software used to realize and optimize the neural network is introduced and the simulated datasets used for training are outlined. This is followed by a short summary of all tunable hyperparameters. Finally a well known problem occurring in machine learning, under- and overtraining, is described and the approach to minimize the effect is discussed.

5.1 Framework

Three software packages are used in this thesis to perform two main tasks, namely training a neural network and optimizing its hyperparameters. Training is performed with a machine learning library called *Pylearn2* [45] which provides all necessary algorithms. The actual computation is performed by *Theano* [46, 47], a compiler on which Pylearn2 is built on top of. The optimization of hyperparameters is performed with a software package called *Spearmint* [48] which is a Bayesian optimization algorithm. The procedure to setup the complete framework is explained in App. B. All packages have been installed on a custom built workstation with the following specifications:

- **CPU:** AMD FX-8320 Octa-Core 3.5 GHz (GigaHertz)
- **GPU:** Nvidia GeForce GTX 770
- **RAM:** 32 GB (GigaByte)

5.1.1 Theano — A math expression compiler

Theano [46, 47] is a compiler for mathematical expressions built to speed up machine learning algorithms in Python. It works with symbolic representations of the mathematical expressions provided by the user. Prior to computation, Theano optimizes the given expression and translates the code either into C++ or CUDA[1]. Since Theano has access to the full computational graph, it can correct slow or unstable numerical expression patterns by performing a local graph transformation. Therefore, it is able to compute expressions such as $\log(1 + e^x)$ even for small values of x numerically correct. It is also able to perform symbolic differentiation which is a crucial part in the stochastic gradient descent algorithm (see Sec. 4.3.4). Algorithms implemented with Theano are $1.6 - 7.5$ times faster on CPU ($6.5 - 44$ times faster on GPU) than competitive implementations in e.g. C/C++ or MATLAB [46].

[1] A programming technique developed by Nvidia to run programs on a *Graphics Processing Unit* (GPU).

5.1.2 Pylearn2 — A machine learning library

Pylearn2 [45] is a machine learning research library built on top of Theano. It is developed at the Université de Montréal by LISA[2]. The philosophy of Pylearn2 is to be flexible and extendable in order to make nearly any machine learning idea feasible. Since it is a research library it is not built for easy use but for easy understanding what each code segment is doing. This often comes with the cost of requiring an experienced machine learning user who understands how the algorithms are working. The main parts of Pylearn2 are the `Dataset`, `Model` and `TrainingAlgorithm` classes. The `Dataset` class stores the data to train on. The `Model` class itself does not perform any numerical computations, but stores parameters concerning e.g. a neural network. For example, it provides the symbolic expression to calculate the forward propagation in a multilayer perceptron (see Sec. 4.3.2). The `TrainingAlgorithm` class performs actual numerical calculations. It takes the parameters from `Model` and adapts them to the given `Dataset`. Each of the three classes has many different subclasses. They can be combined in any way the user wants to build his machine learning model. A possible combination of subclasses to create a deep neural network could look as follows:

- `Dataset`
  - Customized set of input data
- `Model`
  - 5 hidden layers
  - 300 nodes with Tanh activation function in each layer
  - output layer with one node and sigmoid activation function
- `TrainingAlgorithm`
  - SGD training algorithm

5.1.3 Spearmint — A Bayesian optimization algorithm

Machine learning often involves delicate tuning of hyperparameters. Such parameters are for instance the learning rate, the number of hidden layers and the number of nodes per layer. Tuning of these hyperparameters often requires expert experience, rule of thumb or brute force. Therefore, an automated approach which is able to find a good combination of these parameters has a great appeal. Spearmint [48] is a software package to perform Bayesian optimization with Gaussian process priors [49]. As in other optimization problems the focus is on finding the minimum of a given function $f(x)$. In Bayesian optimization a probabilistic model for $f(x)$ is constructed. This model is used to make decisions about which point x of the function $f(x)$ should be evaluated next. The procedure used by Spearmint does not rely on the local information, e.g. a gradient, but on the complete information available from all previous evaluated points. Thus it is able to find the minimum with fewer evaluations of $f(x)$. The drawback of this procedure is that the computation time to decide which point

[2]Laboratoire d'Informatique des Systèmes Adaptatifs
(Informatics Laboratory of Adaptive Systems)

should be next increases with the number of already computed points. In machine learning, however, the evaluation of $f(x)$ is usually time-consuming. The extra computation time is therefore justified by making better decisions in choosing the next point resulting in fewer evaluations of $f(x)$ to reach a minimum.

5.2 Dataset

The 8 TeV and 13 TeV datasets are simulated using the following steps. First, event are generated using the mathematical formalism and models describing the Standard Model. For signal event generation POWHEG [50–54] and PYTHIA6 [55] where used and for background event generation MADGRAPH [56] and PYTHIA8 [55, 57]. The next step is to simulate the detector as well as its response on the generated events which is done with GEANT4 [58]. The last step is the reconstruction of the events according to the information gained from the detector response. The training is performed separately for the 8 TeV and 13 TeV datasets. The 8 TeV dataset corresponds to an integrated luminosity of $\mathcal{L} = 19.7\,\mathrm{fb}^{-1}$ and the 13 TeV dataset to an integrated luminosity of $\mathcal{L} = 10\,\mathrm{fb}^{-1}$. On both datasets a preselection is applied after simulation which sets requirements or cuts on certain quantities (see Sec. 5.2.1). Two differences between the 8 TeV and 13 TeV, concerning the training, are the number of events and the number of input variables, which will be explained in Sec. 5.2.2.

5.2.1 Preselection

Only events are taken where the final state of the $H \to \tau\tau$ decay contains one muon μ and one hadronically decaying tau lepton τ_h. The transverse momenta p_T of the isolated μ and τ_h need to be above 20 GeV. In the VBF production mode the two leading jets originate from quarks (see Fig. 2.3b). It is therefore required that there are at least 2 jets in the final state above p_T thresholds of 45 GeV and 30 GeV, respectively. Furthermore, a minimum η gap of 2.0 is required between the leading jets. A summary including all requirements is listed in Tab. 5.1.

Symbol	Definition	Requirement	Baseline
$p_{\mathrm{T}}(j_\tau)$	isolated τ_{h} above p_{T} threshold	20 GeV	20 GeV
$p_{\mathrm{T}}(\mu)$	isolated muon above p_{T} threshold	20 GeV	20 GeV
$p_{\mathrm{T}}(j_{1,2})$	≥ 2 jets above p_{T} thresholds	45/30 GeV	30/30 GeV
$\Delta\eta(j,j)$	min. η gap between leading jets	2.0	3.5
m_{jj}	min. mass of the two-leading-jets-system	–	500 GeV
m_{vis}	min. mass of μ-τ-system	30 GeV	–
m_{T}	max. transverse mass of μ-$E_{\mathrm{T}}^{\mathrm{miss}}$-system	80 GeV	30 GeV
CJV	central jet veto	–	yes
$q(\mu) \cdot q(\tau)$	product of muon and τ_{h} charge	< 0	< 0
b veto	veto on b-tagged jet in the event	yes	yes
lepton veto	veto on additional lepton in the event	yes	yes

Tab. 5.1: Summary of the preselection. The requirements are looser than for the baseline, which is the preselection of the LHC run-1 analysis, with the exception of two items ($p_{\mathrm{T}}(j_1)$, m_{vis}) which are almost 100% efficient for the signal.

5.2.2 Input variables

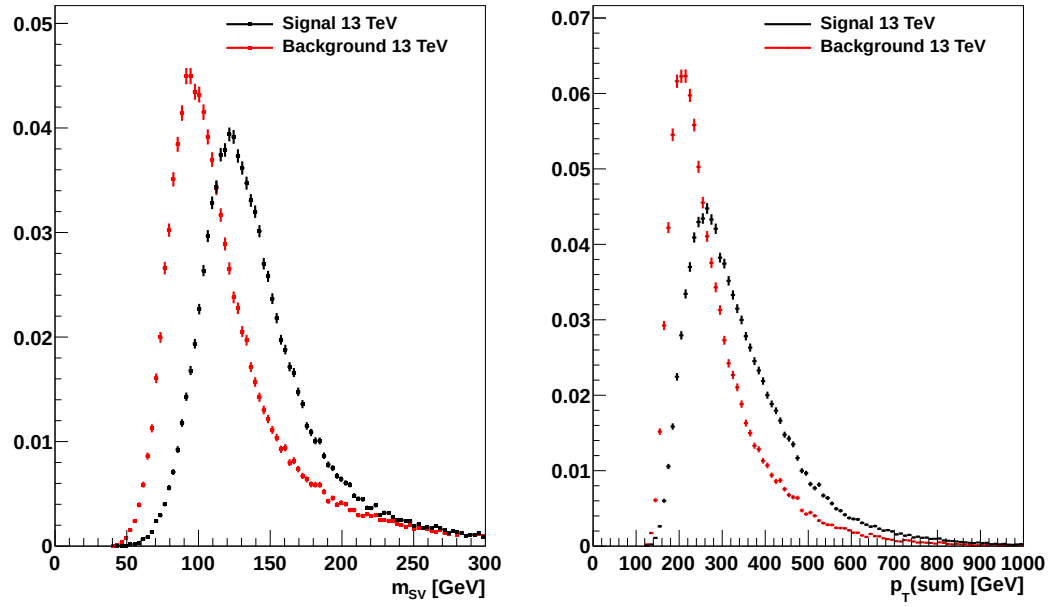
Table 5.2 contains all input variables used to train a deep neural network. All of them are included in the dataset for 8 TeV as well as in the 13 TeV dataset with the exception of $p_{\mathrm{T}}^{\mathrm{sv}}$ which is only included in the former. Therefore, only the distributions of the 12 variables available in both datasets are illustrated in Figs 5.1 – 5.3. The signal events used for training originate from a Higgs boson produced through VBF and decaying to $\tau\tau$ according to the requirements of the previous section. In case of the 8 TeV dataset the events used for testing also include a small amount of events originating from ggF. The background events are from Drell-Yan (in this case $Z \to \tau\tau$) processes. The number of signal and background events in each dataset as well as the weighted number of simulated events are listed in Tab. 5.3. The weighted number of events correspond to the number of events produced by the simulated 8 TeV or 13 TeV collision according to the Standard Model after preselection. The weights of the events are solely used to calculate the Figure of Merit (see Sec. 5.3).

Symbol	Definition
m_{sv}	$m_{\tau\tau}$ estimator SVFit [24]
$\Delta\eta(j,j)$	pseudorapidity gap between leading jets
$p_{\mathrm{T}}(\mathrm{sum})$	scalar p_{T} sum of τ, μ and the two jets
$p_{\mathrm{T}}(\mathrm{tot})$	magnitude of the vector p_{T} sum of τ, μ, the two jets and $E_{\mathrm{T}}^{\mathrm{miss}}$
$\Delta R(\mu,\tau)$	ΔR between μ and τ
m_{vis}	visible mass of μ-τ system
$E_{\mathrm{T}}^{\mathrm{miss}}$ centrality	relative angular $E_{\mathrm{T}}^{\mathrm{miss}}$ position with respect to μ and τ [59]
μ centrality	lepton pseudorapidity with respect to the two jets [59]
m_{jj}	mass of the two-leading-jets system
$\eta_{j1} \cdot \eta_{j2}$	product of the leading jet pseudorapidities
S	sphericity – 3D isotropy of the energy flow
m_{T}	transverse mass of the μ-$E_{\mathrm{T}}^{\mathrm{miss}}$ system
$p_{\mathrm{T}}^{\mathrm{sv}}$ (only 8 TeV)	p_{T} of the $\tau\tau$ resonance estimated by SVFit [24]

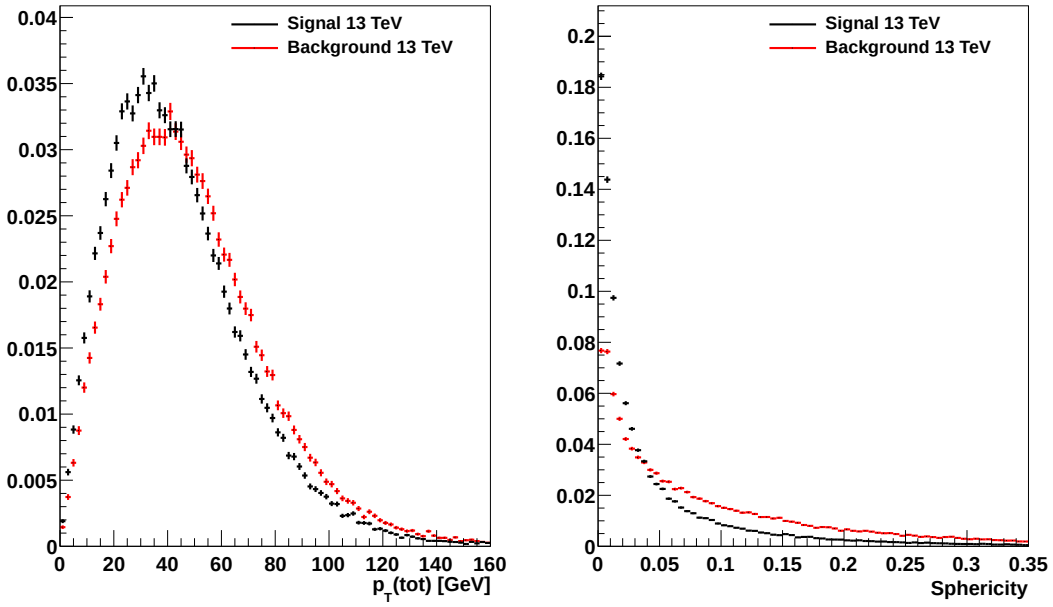
Tab. 5.2: List of observables used as input variables. Here, τ always refers to the reconstructed visible products of a hadronic τ lepton decay.

	8 TeV			13 TeV	
	Test	Train	Weighted	Test + Train	Weighted
Signal	5449 + (669)	5446	35	89764	22
Background	5899	5916	1420	78404	960

Tab. 5.3: The number of signal events in the 8 TeV test dataset include, beside the VBF events, a small amount of ggF (number written in brackets). All other signal events are exclusively from VBF. The weighted events are the total signal or background events multiplied with a weight factor respectively.

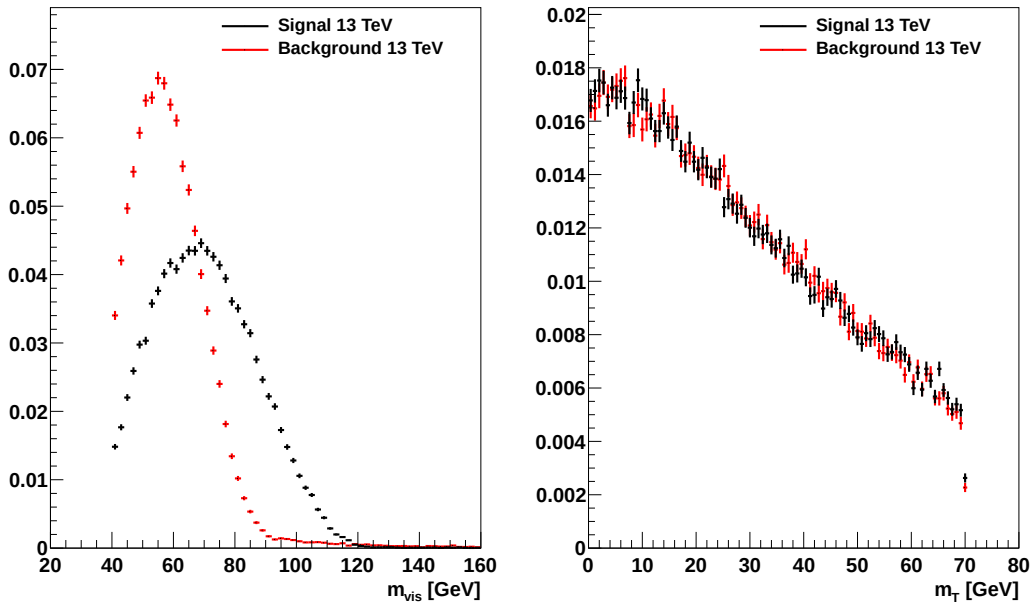


(a) m_{sv} (b) $p_{\mathrm{T}}(\mathrm{sum})$

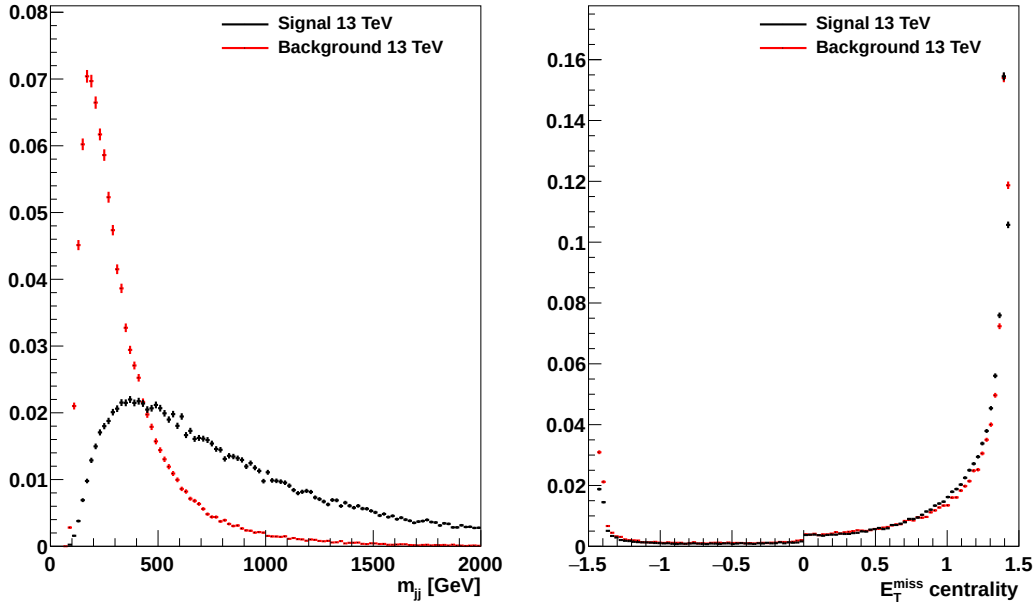


(c) $p_{\mathrm{T}}(\mathrm{tot})$ (d) Sphericity S

Fig. 5.1: Distributions of the variables m_{sv}, $p_{\mathrm{T}}(\mathrm{sum})$, $p_{\mathrm{T}}(\mathrm{tot})$ and Sphericity S [60].

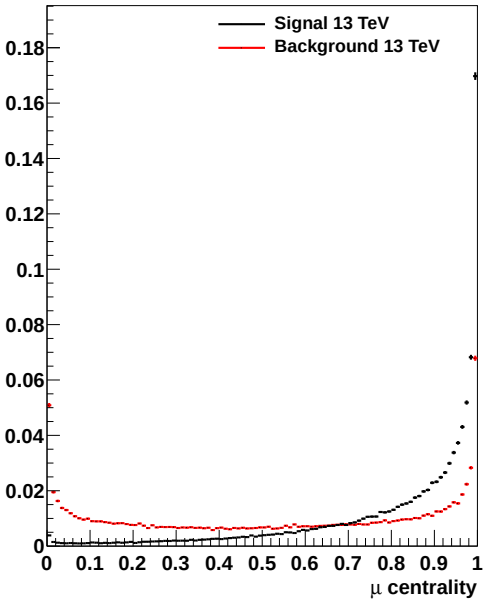


(a) m_{vis} (b) m_{T}

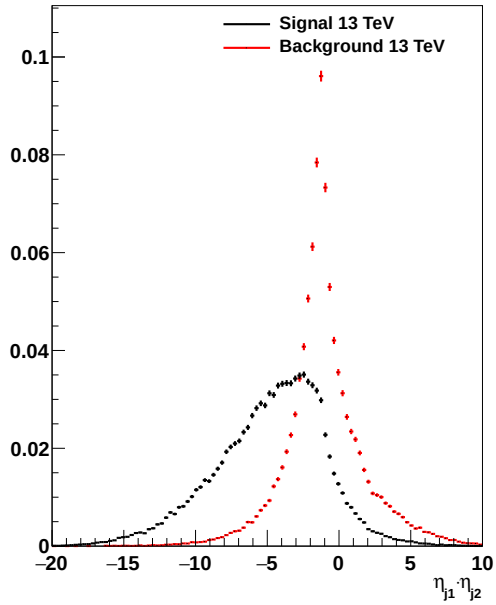


(c) m_{jj} (d) $E_{\mathrm{T}}^{\mathrm{miss}}$ centrality

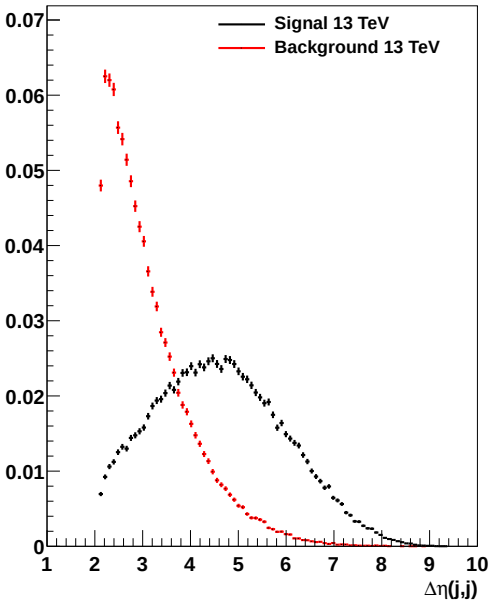
Fig. 5.2: Distributions of the variables m_{vis}, m_{T}, m_{jj} and $E_{\mathrm{T}}^{\mathrm{miss}}$ centrality [60].



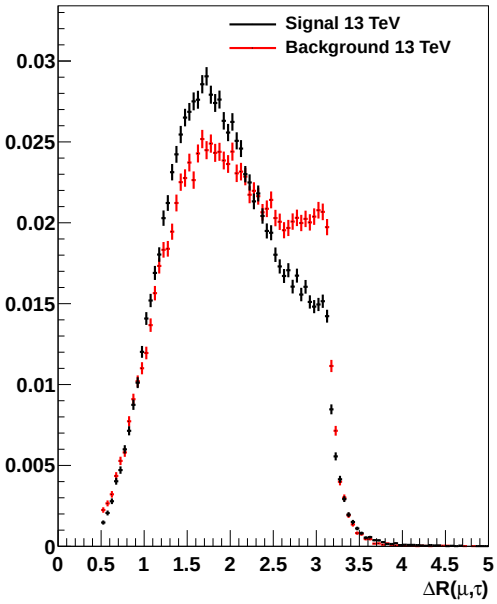
(a) Lepton centrality



(b) $\eta_{j1} \cdot \eta_{j2}$



(c) $\Delta\eta(j, j)$



(d) $\Delta R(\mu, \tau)$

Fig. 5.3: Distributions of the variables μ centrality, $\eta_{j1} \cdot \eta_{j2}$, $\Delta\eta(j, j)$ and $\Delta R(\mu, \tau)$ [60].

5.3 Figure of Merit - Approximate Median Significance

While the loss function is sufficient to compare the performance of different neural networks it is not very useful when comparing the neural network with other machine learning approaches. Therefore, the *Figure of Merit* (FoM) chosen to achieve comparability between different approaches is the *Approximate Median Significance* (AMS)

$$\mathrm{AMS}(s,b) = \sqrt{2\left[(s+b+b_{reg})\log\left(1+\frac{s}{b+b_{reg}}\right)-s\right]}, \tag{5.1}$$

where s is the number of signal-like events and b is the number of background-like events. The regularization term b_{reg} is a constant and prevents the denominator in the log-function from becoming too small for very few background-like events which would lead to an overestimation of the AMS value. Equation 5.1 is a simplified version since no statistical or systematic errors are included. Nevertheless, the AMS evaluation in the analysis uses the complete computation.

The first step to calculate the FoM is to train a neural network on the training dataset. This neural network is then used to classify the examples given in the test dataset. Each example in the test dataset is labeled as either signal or background event accordingly. The output value of each network used in this thesis is a real number between 0 – 1, where 0 corresponds to a background-like event and 1 corresponds to a signal-like event. The closer the output value is to 0 or 1 the more likely it is for this specific example to be a background or signal event respectively. This value is used to fill one of two histograms, one for the examples labeled as background and one for the examples labeled as signal. The number of events b in the background histogram and the number of events s in the signal histogram corresponding to the bin with the same value in each histogram is then used to calculate the AMS according to eq. 5.1. The FoM is then calculated by adding the AMS value of each bin quadratically.

5.4 Tunable hyperparameters

In this section a short summary of all tunable hyperparameters is given. In addition, some parameters chosen to be constant are included in order to get a complete parameter list of the deep neural network used in the following chapters.

5.4.1 Network architecture

The tunable parameters for the network architecture are hidden layers and nodes. The output layer is the same for all deep neural networks trained in this thesis and uses one node with a sigmoid activation function. Furthermore, the number of nodes is equal in each hidden layer. In Chap. 6 only the hyperbolic tangent activation function is used. In Chap. 7 two additional activation functions are used, the rectified linear and the rectified log activation function (see Sec. 7.1.3).

- **Number of hidden layers**

  The number of hidden layers can take all positive integer values. Computation time is mostly determined by this value. Its influence on the performance of the neural network is discussed in Sec. 6.2.3.

- **Number of nodes**

  The number of nodes can, as the number of hidden layers, take all positive integer values. The nodes in combination with the connection weights are responsible for learning the information provided by the training input.

5.4.2 Training algorithm

In this thesis, the algorithm chosen to train a given network architecture is minibatch SGD with decaying learning rate and a momentum term. The training is either stopped after a fixed number of iterations or if the training error starts to increase. Reasons for a increasing training error are a bad configuration of hyperparameters or overtraining (see Sec. 5.4.3).

- **Minibatch size**

  For the use of minibatch stochastic gradient descent (see Sec. 4.3.4) the size of the minibatch needs to be defined. It is chosen to have a constant value of 100 since a change in minibatch size requires a change of the learning rate. Therefore, results achieved with different batch sizes are only comparable in network performance but not in the chosen hyperparameter combinations. Before each training iteration of the minibatch SGD algorithm the training set is permuted and then split into minibatches. The training of the neural network using all minibatches once, which is equivalent to the complete training-set, is called one training epoch.

- **Number of training epochs**

  Each epoch the gradient of all minibatches is calculated and used to update the weights and biases. If the number of training epochs is too small the training will stop before reaching a minimum. If the number is too large the training will start to overfit. In this thesis a upper threshold of epochs is chosen. In order to prevent the training from overfitting early stopping is used, which is discussed in Sec. 5.4.3.

- **Learning rate**

  The learning rate is a real positive number. It determines how fast the learning algorithm (such as SGD) converges to a minimum and is responsible for the quality of this minimum. The performance of a neural network with different combinations of learning rate and decay factor is discussed in Sec. 6.1.1

- **Decay factor of the learning rate**

  With a decaying learning rate it is possible to choose a rather large initial learning rate. This leads to a fast progressing learning algorithm at the beginning

of training. A small learning rate towards the end ensures that a more precise value of the minimum is reached. The decay factor is a real positive number and always larger than 1. The learning rate is divided by this factor after each epoch yielding an exponential decay. To prevent the learning rate from decaying to 0 a minimal learning rate of 10^{-7} is chosen.

- **Initial and final value of the momentum**

  The initial and final values of the momentum are real numbers between 0 and 1. A standard choice is a momentum increasing from 0.5 to 0.99.

- **Number of epochs from initial to final momentum value**

  It determines how fast the momentum should increase from the initial to the final value.

Fine tuning of the three momentum parameters (inital momentum, final momentum (FM) and number of epochs) in combination with the learning rate and its decay factor leads to the best possible result for a given network architecture.

5.4.3 Overtraining

Prior to optimizing a neural network it is useful to determine how long one has to train a neural network to obtain the best possible result. For a non-ideal training time there are two possible outcomes, undertraining and overtraining. Undertraining (see Fig. 5.4) occurs when the time is too short for the neural network to learn the underlying model of a given training dataset. Overtraining, on the other hand, occurs when the training time is too long and the neural network learns the examples in the training dataset "by heart". Although the overtrained neural network classifies the training dataset nearly perfectly it performs poorly on slightly different datasets such as the test dataset. The first step to prevent under- and overtraining is to split the training dataset into a validation set and a training set. The validation set is used to monitor the classification error during training and contains about $10 - 20\%$ of the examples from the training dataset. The training set contains the rest of the examples and is used, as the name already suggests, to train the neural network. As long as the classification error on the validation set is decreasing training is continued in order to prevent undertraining. This corresponds to the steep decreasing region at the beginning of Fig. 5.5. To prevent overtraining a simple method is to use "early stopping". This method monitors the training progress and stops training as soon as the classification error is not improving further. The interesting region for early stopping is the one denoted with "optimum" in Fig. 5.5. In this region the classification error is only decreasing slowly which indicates that the training is near the best possible result and can be stopped. In the presented example it is best to perform training between 50 and 800 epochs to avoid under- and overtraining, respectively.

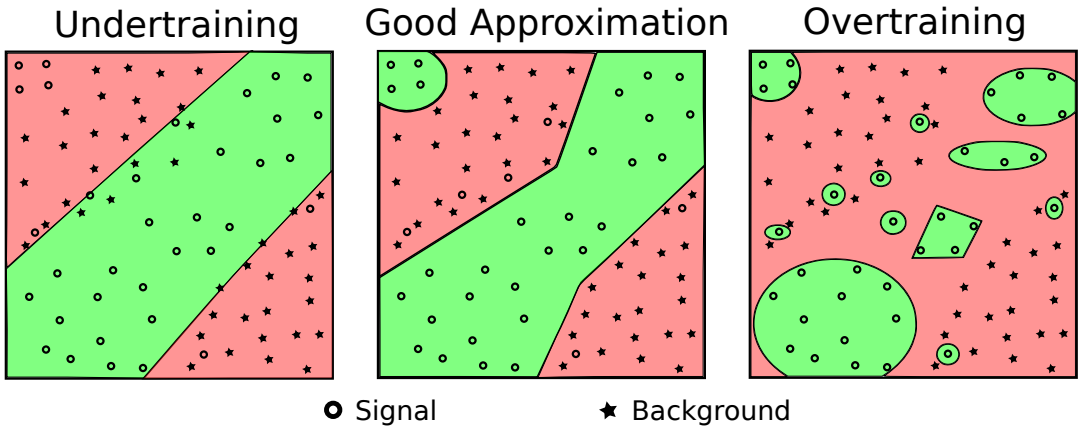


Fig. 5.4: Decision boundaries for a under- and overtrained neural network as well as for a network trained to a optimum.

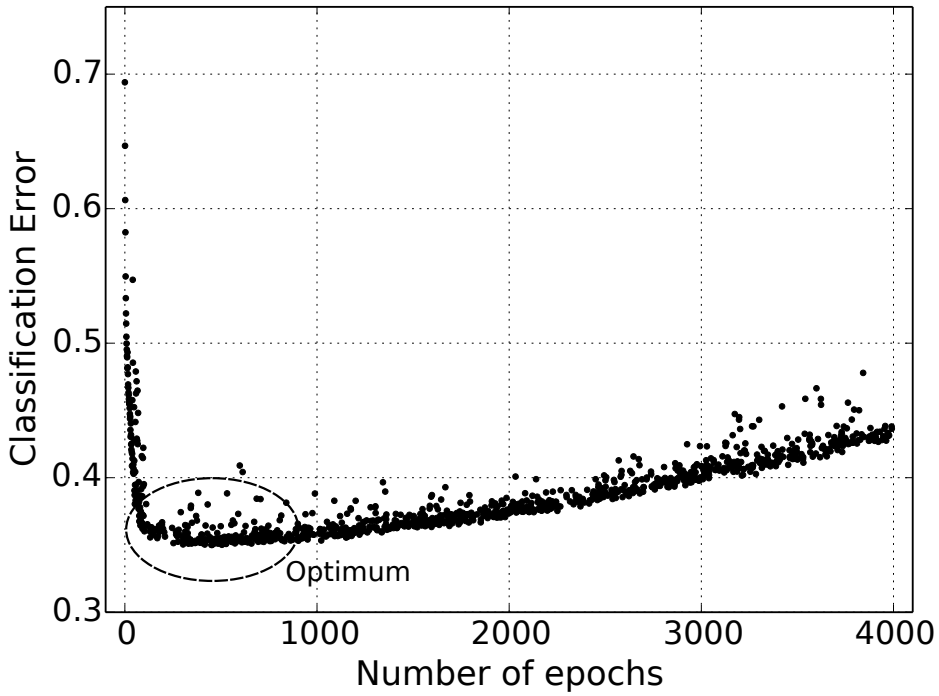


Fig. 5.5: Classification error depending on the number of epochs and therefore on the training time.

6 Multivariate analysis of $H \to \tau\tau$ using the simulated 8 TeV dataset

In this chapter the hyperparameters of a deep neural network realized with Pylearn2 are studied by using the simulated 8 TeV dataset. For this purpose the parameters are optimized with two different approaches. The first approach is to use grid search followed by an optimization algorithm. Using grid search, different learning rate and decay factor combinations are used on several neural network architectures followed by the inclusion of a momentum term in the learning algorithm. The optimization algorithm is used to perform the same task and is compared to the grid search approach afterwards. Finally, the best subset of input variables is selected and the influence of different numbers of hidden layers is discussed.

6.1 Training deep neural networks using grid search

Without expert knowledge in training neural networks one possible choice is to use grid search in order to find good combinations of hyperparameters. In Sec. 5.4 a total list of 7 tunable hyperparameters, consisting of 2 parameters for the architecture of the network and 5 parameters for the learning algorithm, is given. Thus, one has to solve a 7-dimensional optimization problem with grid search.

6.1.1 Combinations of learning rate and decay factor

For this purpose the training is performed with a decaying learning rate only. Therefore, the values for the initial and final momentum are set to 0. The idea is to determine the network performance for different combinations of learning rate and decay factor by performing a grid search. The minimum and maximum values for the learning rate as well as for the decay factor used for grid search are shown in Tab. 6.1. These intervals are split logarithmically equidistant into 20 points resulting in a grid of 400 combination overall. All learning rate and decay factor combinations are calculated for neural networks with 3 or 5 hidden layers with either 300 or 500 nodes in each layer yielding 4 different network architectures. In Fig. 6.1a and 6.1b the results for the neural network with 3 hidden layers and in Fig. 6.2a and 6.2b the results for the

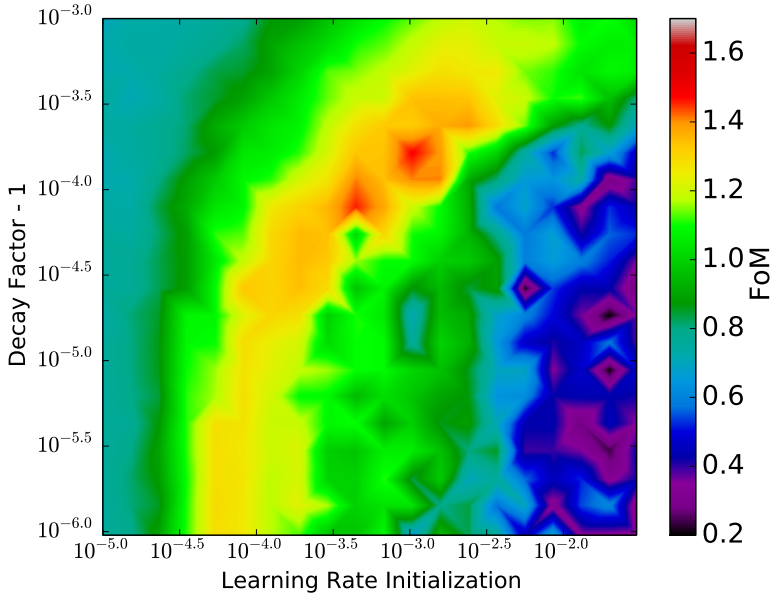
	Minimum	Maximum
Learning Rate	10^{-5}	$3 \cdot 10^{-2}$
Decay Factor	$1 + 10^{-6}$	$1 + 10^{-3}$

Tab. 6.1: Parameter space of the learning rate and the decay factor for grid search.

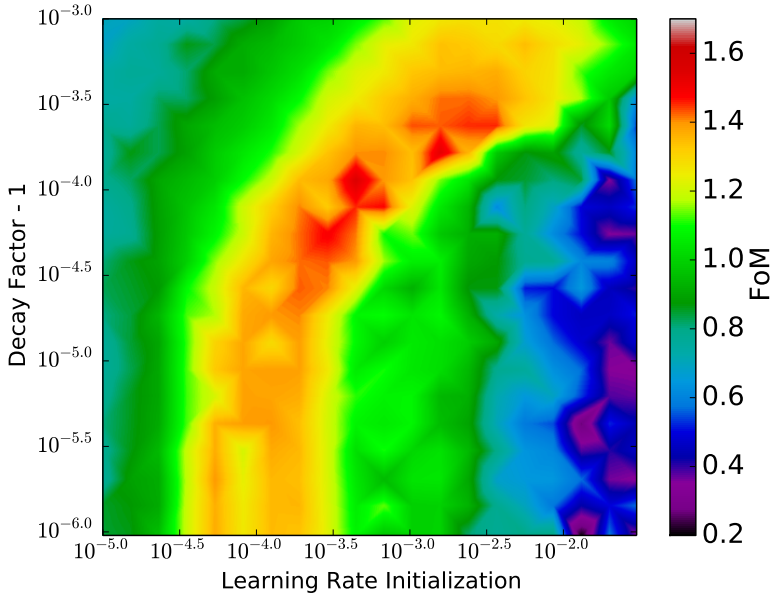
neural network with 5 hidden layers are shown. By comparing the performance for different learning rate and decay factor combinations of all four network architectures, it appears that the region for a learning rate $> 10^{-2.5}$ is dominated by a low FoM. For a learning rate between $10^{-4.5}$ and $10^{-2.5}$ the highest FoM can be found. The influence of the decay factor is mostly dominant in this region yielding the curved shape of the high FoM region visible in all four figures. Summarizing one can say that after performing the grid search all four network architectures show nearly the same behavior for all combinations of learning rate and decay factor. Looking at the FoM (see Tab. 6.2) the biggest difference can be found between the networks with 3 and 5 hidden layers, the network with 5 hidden layers performing better than the network with 3 hidden layers. In the network with 3 hidden layers the number of nodes leads to slightly different FoM while in the network with 5 hidden layers the FoMs are almost equal. The learning rate and decay factor values are similar for the best neural networks. The FoM therefore depends mostly on the number of hidden layers and the number of nodes.

Hidden Layers	Nodes	Learning Rate	Decay Factor	FoM
3	300	10^{-3}	$1 + 2 \cdot 10^{-4}$	1.48
3	500	$4 \cdot 10^{4}$	$1 + 10^{-4}$	1.55
5	300	10^{-3}	$1 + 10^{-4}$	1.67
5	500	10^{-3}	$1 + 2 \cdot 10^{-4}$	1.68

Tab. 6.2: Highest FoM obtained after performing a grid search for different learning rate and decay factor combinations. The deep neural networks used 3 or 5 hidden layers with either 300 or 500 nodes in each hidden layer.

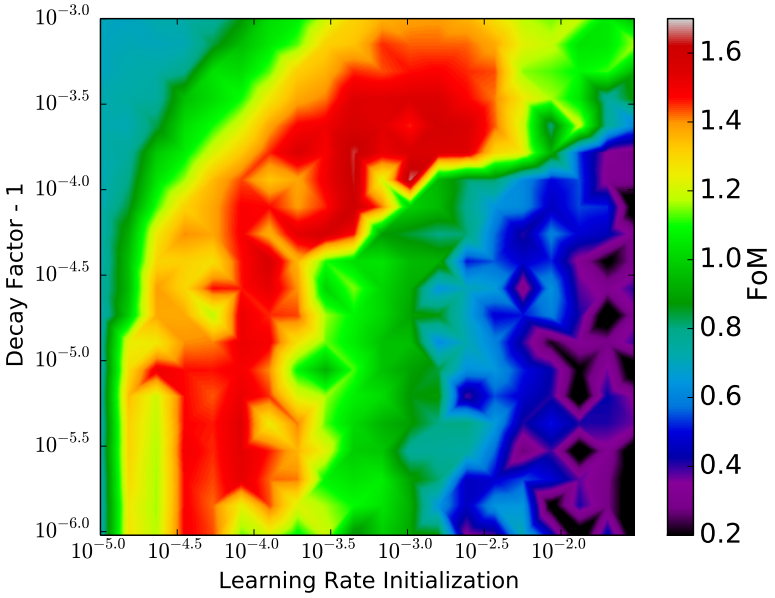


(a) 300 nodes.

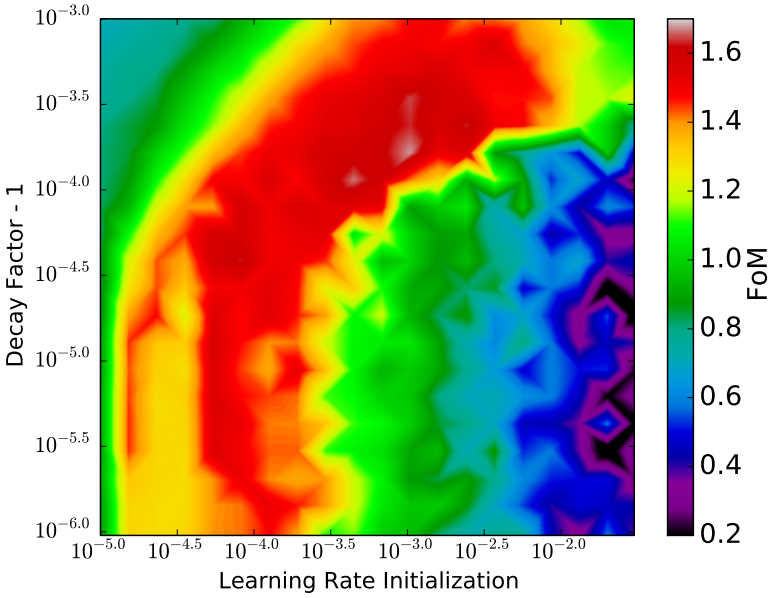


(b) 500 nodes.

Fig. 6.1: FoMs for a grid search using different learning rate and decay factor combinations. The deep neural networks used 3 hidden layers and either 300 or 500 nodes in each layer.



(a) 300 nodes.



(b) 500 nodes.

Fig. 6.2: FoMs for a grid search using different learning rate and decay factor combinations. The deep neural networks used 5 hidden layers and either 300 or 500 nodes in each layer.

6.1.2 Combinations of learning rate and decay factor with momentum

Similar to Sec. 6.1.1 the optimization is performed with a grid search. The main difference is that a momentum term is included in the training algorithm. The neural network architecture chosen to train with the additional momentum term is the one with the best result in the previous section (see Tab. 6.2) and has 5 hidden layers and 500 nodes. The momentum extension of the SGD training algorithm is determined by three hyperparameters as discussed in Sec. 5.4. The momentum is chosen to increase from 0.5 to 0.99. The values for the initial and final momentum are constant during this section since more combinations would only result in a much longer computation time without providing more information on the influence of the momentum term. The grid for the learning rate and decay factor is chosen equal to the previous section (see Tab. 6.3). In Fig. 6.3 and 6.2 the results for a momentum increasing from the initial to the final value after either 20, 50, 100 or 200 epochs are shown. Similar to the results

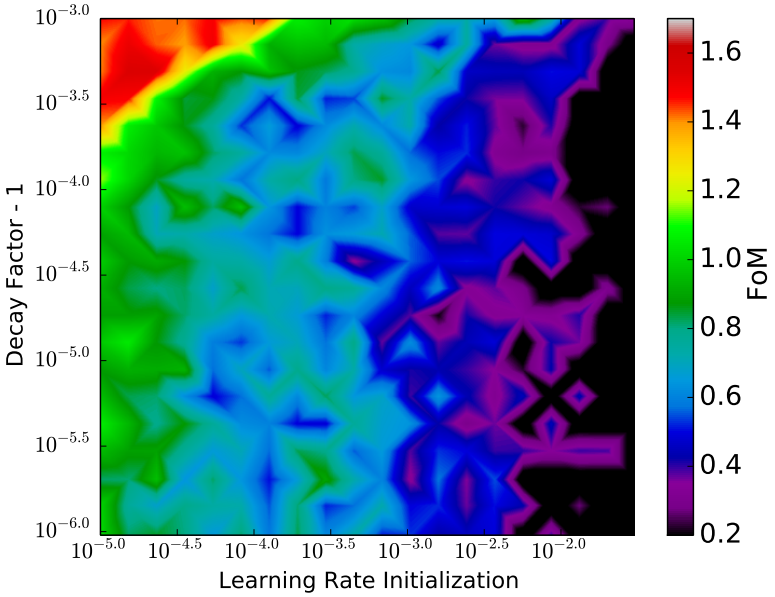
	Minimum	Maximum
Learning Rate	10^{-5}	$3 \cdot 10^{-2}$
Decay Factor	$1 + 10^{-6}$	$1 + 10^{-3}$
Initial Momentum	0.5	0.5
Final Momentum (FM)	0.99	0.99
Epochs to FM	20	200

Tab. 6.3: Parameter space of the hyperparameters used for grid search.

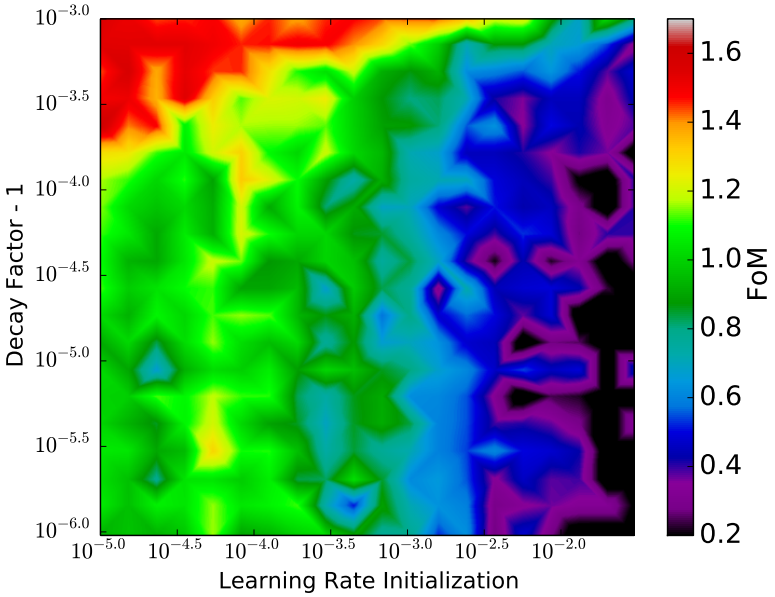
obtained without momentum (see Fig. 6.1 and Fig. 6.2) the region with a learning rate $> 10^{-2.5}$ is dominated by a low FoM and is independent of the number of epochs after which the momentum reaches the final value. Figures 6.3a and 6.3b favor a fast increasing momentum yielding an aggressive training behavior. Therefore, a high FoM is only obtained when the initial learning rate is already small and decays fast to the absolute minimum value. In this case, training is dominated by the momentum term. For a slower increasing momentum, as in Fig. 6.4a and 6.4b, learning rate and momentum are equally important for training leading to a similar shape of the high FoM region as in the previous section. Nevertheless, the maximum FoM is only slightly higher than for the training without momentum (compare Tab. 6.2 and Tab. 6.4).

Learning Rate	Decay Factor	Momentum saturation	FoM
$1.5 \cdot 10^{-5}$	$1 + 5 \cdot 10^{-4}$	20	1.56
10^{-6}	$1 + 2 \cdot 10^{-4}$	50	1.61
$3 \cdot 10^{-4}$	$1 + 3 \cdot 10^{-4}$	100	1.62
$1.5 \cdot 10^{-5}$	$1 + 10^{-3}$	200	1.70

Tab. 6.4: Highest FoM obtained after performing a grid search for different learning rate and decay factor combinations. The deep neural networks used 5 hidden layers and 500 nodes in each hidden layer. The momentum term increased from 0.5 to 0.99 after either 20, 50, 100 or 200 epochs.

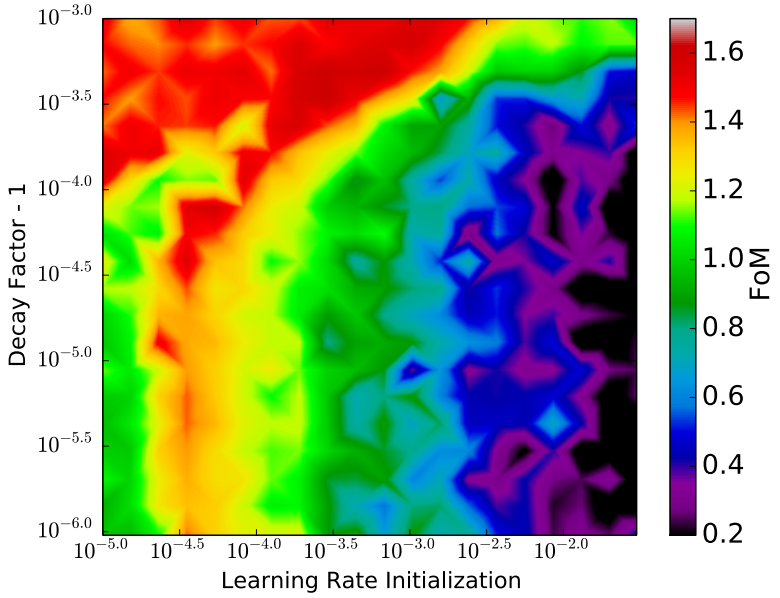


(a) Momentum saturating after 20 epochs.

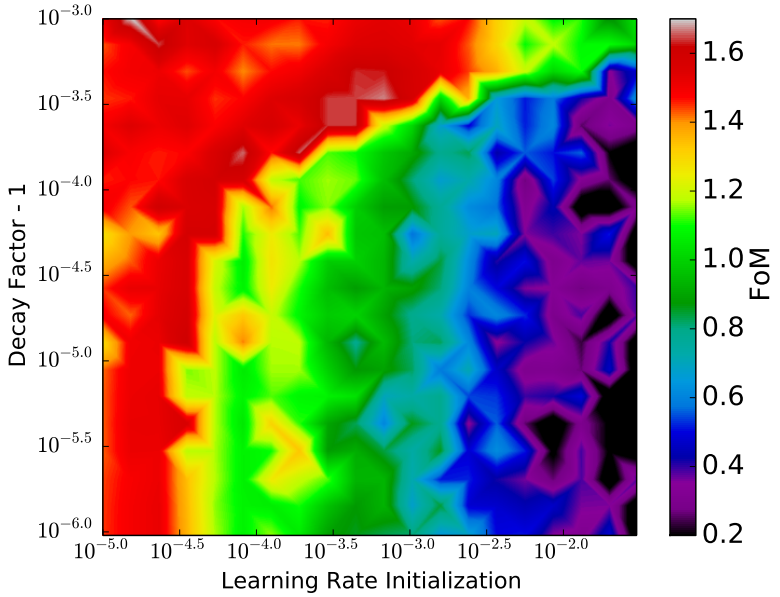


(b) Momentum saturating after 50 epochs.

Fig. 6.3: FoMs for a grid search using different learning rate and decay factor combinations. The deep neural networks used 5 hidden layers and 500 nodes in each layer. The momentum term increased from 0.5 to 0.99 after either 20 or 50 epochs.



(a) Momentum saturating after 100 epochs.



(b) Momentum saturating after 200 epochs.

Fig. 6.4: FoMs for a grid search using different learning rate and decay factor combinations. The deep neural networks used 5 hidden layers and 500 nodes in each layer. The momentum term increased from 0.5 to 0.99 after either 20 or 50 epochs.

6.2 Training deep neural networks using Spearmint

The bottleneck of optimization with grid search as performed in Sec. 6.1 is the computation time for each combination of hyperparameters. In order to find the global maximum the density of points and the dimension of the grid to evaluate in a grid search has to be high, resulting in a computation time in the order of several days. Therefore, a favorable choice is to use a sophisticated optimization algorithm which is able to find a good combination of hyperparameters in a shorter time. The optimization algorithm used in this thesis is called Spearmint (see Sec. 5.1.3)

6.2.1 Necessary Spearmint iterations to reach optimum

The most interesting information about an optimization algorithm is the number of iterations required to a maximum and the quality of this maximum. To obtain this information a neural network with 5 hidden layers and 500 nodes is optimized with Spearmint. As training algorithm a decaying learning rate with a momentum term using the same minimum and maximum values as in Sec. 6.1.2 is used. Spearmint is allowed to perform 200 iterations during which all parameters listed in Tab. 6.5 are optimized at once. The optimization process is repeated 5 times with differently initialized weights in order to monitor the optimization process and not the training for a specific initialization. Figure 6.5 shows how the FoM is increasing after a certain

	Minimum	Maximum
Learning Rate	10^{-5}	$3 \cdot 10^{-2}$
Decay Factor	$1 + 10^{-6}$	$1 + 10^{-3}$
Initial Momentum	0.01	0.8
Final Momentum	0.5	0.99
Epochs to FM	20	200

Tab. 6.5: Parameter space of the hyperparameters used to optimize with Spearmint.

number of iterations. The steepest ascend can be found between 1 and 50 iterations and the maximum FoM is reached after 128 iterations. However, a value close to the maximum was already found after about 60 iterations. Summarizing one can say that Spearmint needs at least 50 iterations to get near the maximum value but is only slightly improving afterwards. In Sec. 6.1.2 four different values of epochs to reach the final momentum were used in a grid of 400 learning rate and decay factor combinations yielding 1600 points to compute. To obtain the best FoM with grid search a much longer computation time was necessary than with Spearmint (see Tab. 6.6) even though the parameter space for learning rate, decay factor and epochs to reach the final momentum was chosen equal and the dimension of the optimization problem was increased by adding the initial and final momentum to the parameter space when using Spearmint. The higher FoM obtained with Spearmint is mostly due to the fact that the values for initial and final momentum were also optimized.

	Grid Search	Spearmint
Hidden Layers	5	5
Nodes	500	500
Learning Rate	$1.5 \cdot 10^{-5}$	10^{-4}
Decay Factor	$1 + 10^{-3}$	$1 + 1.8 \cdot 10^{-4}$
Initial Momentum	0.5	0.8
Final Momentum	0.99	0.99
Epochs to FM	200	200
FoM	1.70	1.76

Tab. 6.6: Comparison of the highest FoM and the hyperparameters used for grid search and Spearmint.

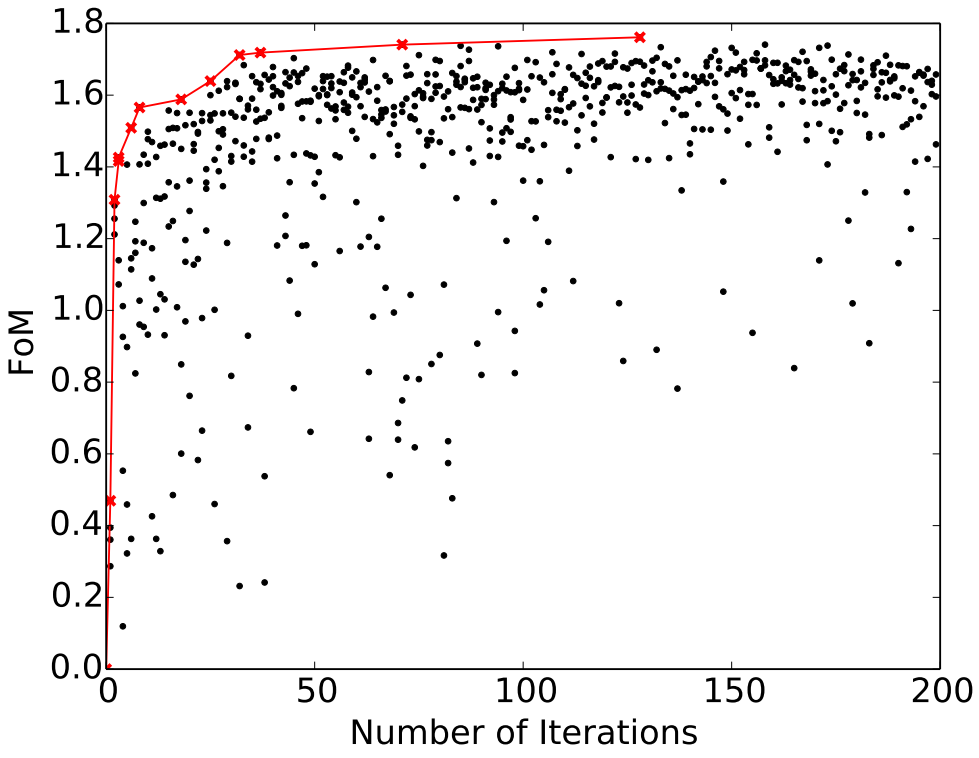


Fig. 6.5: Number of iterations after which Spearmint reaches a maximum. The red crosses represent the current maximum value during the optimization. The black points represent the FoM obtained at a certain iteration step.

6.2.2 Input variable selection

In Sec. 6.1 – 6.2.1 the complete set of input variables, as listed in Tab. 5.2, is used. This ensemble of variables is chosen to contain highly discriminative features to distinguish between signal and background. However, for some analysis techniques it is possible that a smaller subset of this ensemble yields better results [61] due to the fact that irrelevant and redundant input variables are removed. The effect of redundant variables is to increase the number of local optima during training since there are more combinations of parameters that can yield locally optimal training results. Irrelevant variables add noise to the input and therefore hinder the training process. A straight forward approach of finding the best subset would be to try all possible combinations. A computationally more feasible way is to iteratively remove one feature starting with the complete set of N variables. In the first iteration step N combinations of $N-1$ variables are computed. The combination of $N-1$ variables obtaining the best result is the starting point for the next iteration step. This time $N-1$ combinations of $N-2$ variables are computed and as before the combination yielding the best result is kept. This is repeated until only one variable is left. This feature gets the highest rank since it has the most discriminative power, whereas the feature already removed in the first step gets the lowest rank. While trying all possible combinations from 1 to 13 input features would take about 8200 iterations, the second approach is about 90 times faster.

In Fig. 6.6 the FoM as a function of the number of variables is plotted, and in Tab. 6.7 the ranking after selection of the input variables available in the 8 TeV dataset is listed. The result is that a subset of 9 variables yields a significant improvement of the FoM compared to the result obtained with all variables.

Rank	Feature
1	m_{sv}
2	p_{T}^{sv}
3	μ centrality
4	S
5	$p_{\text{T}}(\text{tot})$
6	m_{T}
7	$\Delta\eta(j,j)$
8	m_{vis}
9	$\Delta R(\mu,\tau)$
10	$\eta_{j1}\cdot\eta_{j2}$
11	$p_{\text{T}}(\text{sum})$
12	$E_{\text{T}}^{\text{miss}}$ centrality
13	m_{jj}

Tab. 6.7: Ranking of the input variables where 1 is the highest and 13 is the lowest rank.

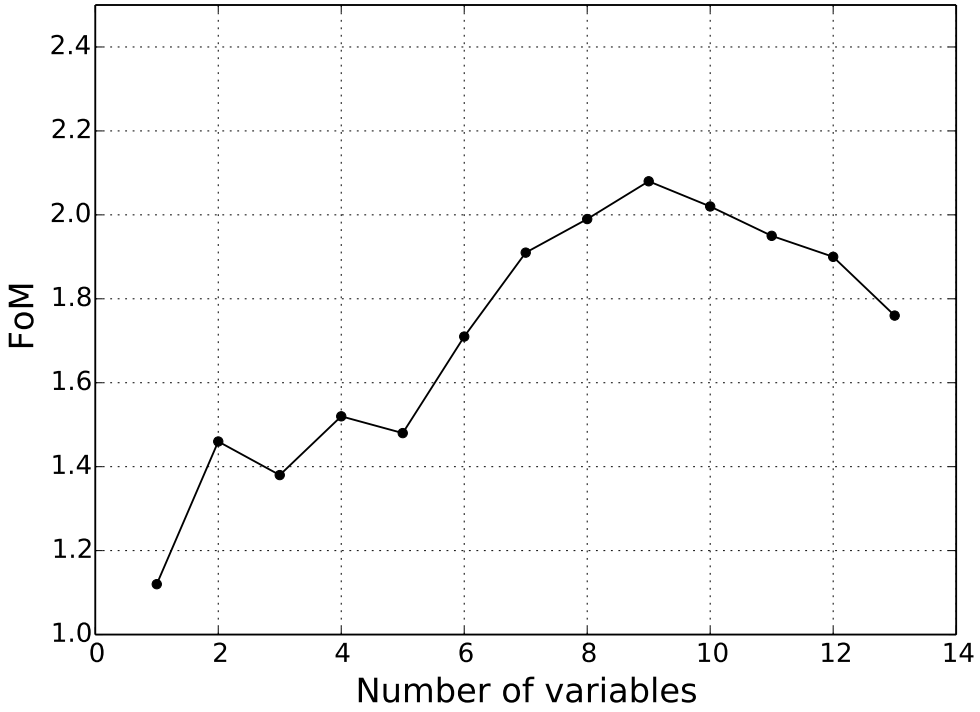


Fig. 6.6: Highest FoM as a function of the number of input variables for the 8 TeV dataset.

6.2.3 Performance of neural networks with multiple hidden layers

In Sec. 6.1 – 6.2.2 the upper and lower limits of the training algorithm hyperparameters as well as the best combination of input variables are explored. The final hyperparameter to potentially improve the performance of a neural network, the number of hidden layers, is discussed here. Neural networks with different numbers of hidden layers, ranging from 1 to 8, are optimized with Spearmint and compared afterwards. The hyperparameters used to optimize these networks are listed in Tab. 6.8. In Fig. 6.7

	Minimum	Maximum
Nodes	100	700
Learning Rate	10^{-5}	$3 \cdot 10^{-2}$
Decay Factor	$1 + 10^{-6}$	$1 + 10^{-3}$
Initial Momentum	0.01	0.8
Final Momentum	0.5	0.99
Epochs to FM	20	200

Tab. 6.8: Parameter space of the hyperparameters used to optimize with Spearmint. The neural networks use 1 to 8 hidden layers.

the best result after optimizing the hyperparameters for a specific number of hidden layers is illustrated. The performance improves considerably for each additional hidden layer up to seven hidden layers and decreases slightly for eight hidden layers. A possible explanation why the FoM is not increasing further when using more than 7 hidden layers is the vanishing gradient problem (see Sec. 4.4).

To overcome the vanishing gradient the training has to be performed longer for every additional layer since the backpropagated error decays exponentially [43]. Longer training can be achieved either by training more epochs when using a "small" dataset or by using a "large" dataset since one epoch is defined as one iteration over the complete dataset. Thus, training with a small dataset takes more epochs than training with a large dataset to achieve a comparable result. Training with a small dataset for too many epochs will result in overtraining yielding the conclusion that the size of the dataset is a limiting factor for the performance of a deep neural network. Therefore, using a large dataset is essential when training a DNN. However, what dataset is too small and what dataset is large enough depends strongly on the given problem.

Since the training algorithm uses early stopping, as introduced in Sec. 5.4.3, to detect overtraining, a decrease in performance when using more than 7 hidden layers suggests that these deeper network starts to become sensitive to overtraining because of a too small dataset size and is therefore stopped early. The highest FoM depending on the number of hidden layers and the optimized hyperparameters for the 8 TeV dataset are listed in Tab. 6.9.

Hidden Layers	1	2	3	4
Nodes	413	500	500	500
Learning Rate	10^{-5}	10^{-5}	10^{-5}	$9 \cdot 10^{-4}$
Decay Factor	$1 + 4.4 \cdot 10^{-6}$	$1 + 10^{-6}$	$1 + 1.4 \cdot 10^{-6}$	$1 + 10^{-3}$
Initial Momentum	0.47	0.44	0.08	0.66
Final Momentum	0.84	0.50	0.50	0.94
Epochs to FM	200	20	63	20
FoM	1.63	1.68	1.74	1.75

Hidden Layers	5	6	7	8
Nodes	500	500	500	500
Learning Rate	$1.2 \cdot 10^{-4}$	$3.4 \cdot 10^{-4}$	$1.4 \cdot 10^{-4}$	10^{-4}
Decay Factor	$1 + 1.7 \cdot 10^{-4}$	$1 + 2.7 \cdot 10^{-4}$	$1 + 3.5 \cdot 10^{-4}$	$1 + 2 \cdot 10^{-4}$
Initial Momentum	0.80	0.01	0.50	0.50
Final Momentum	0.81	0.65	0.78	0.80
Epochs to FM	46	26	200	200
FoM	1.80	1.92	2.04	1.99

Tab. 6.9: Highest FoM and optimal hyperparameters for neural networks with 1 to 8 hidden layers.

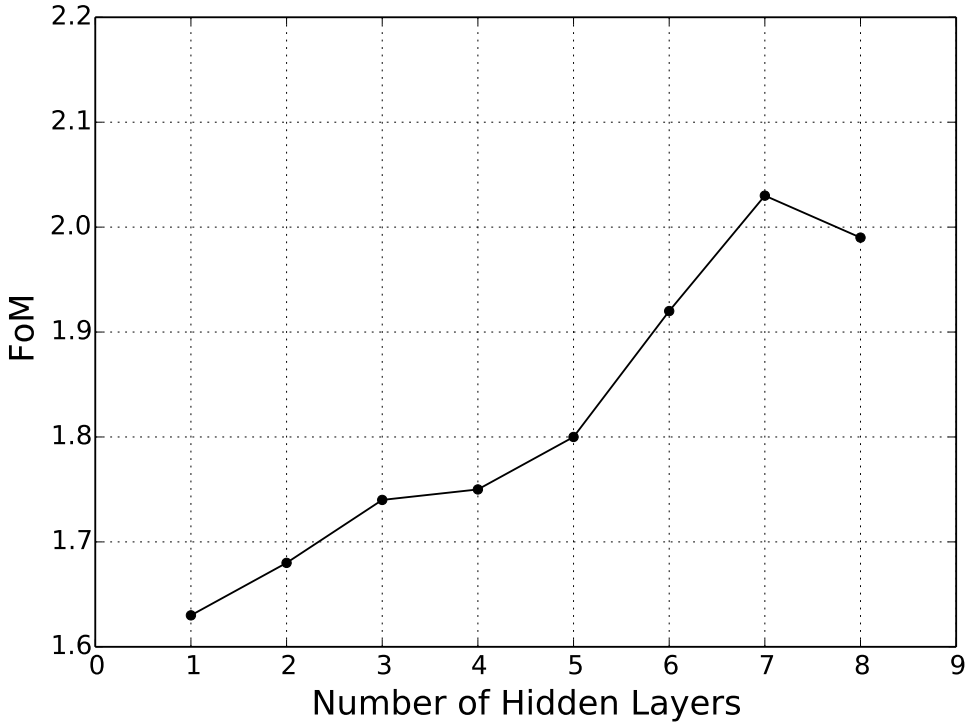


Fig. 6.7: Improvement of the FoM when using more hidden layers.

6.2.4 Preprocessing of input variables

For many machine learning algorithms it is necessary that the input variables have the same range, e.g. values between 0 and 1. This guarantees that the training is not distorted by large input values [61]. The distributions of the input variables listed in Sec. 5.2.2 have different minimum and maximum values. Therefore, the last optimization step for the classification of the 8 TeV dataset is to subtract the mean of the input variable distribution from each variable respectively and scale it to have unit variance. Centering and scaling is done independently for each variable after computing the relevant statistics on the events in the training set. In Tab. 6.10 the hyperparameters of the deep neural network with the highest FoM are listed. Due to the preprocessing, the FoM improved by 0.1 compared to the highest FoM of the deep neural network in Sec. 6.2.3. Another consequence of this preprocessing is that fewer nodes are required in each layer compared to neural networks trained on the unscaled input variables.

Hyperparameter	Value
Hidden Layers	7
Nodes	20
Learning Rate	$2 \cdot 10^{-3}$
Decay Factor	$1 + 10^{-6}$
Initial Momentum	0.50
Final Momentum	0.99
Epochs to FM	20
FoM	2.14

Tab. 6.10: Highest FoM and optimal hyperparameters for the 8 TeV dataset when using preprocessed input variables.

In Fig. 6.8 the distribution of the signal and background events after classification by the deep neural network is shown. Since there is only one node with a sigmoid activation function in the output layer the output of the DNN ranges from 0 to 1, where 0 corresponds to background-like events and 1 to signal-like events. The FoM is calculated by adding the AMS values of each bin quadratically. The biggest contribution to the FoM is made by the last bin because of the excess of expected signal events compared to the background events. The b_{reg} term in eq. 5.1 is set to $b_{reg} \approx 0$ for the computation of the final FoM.

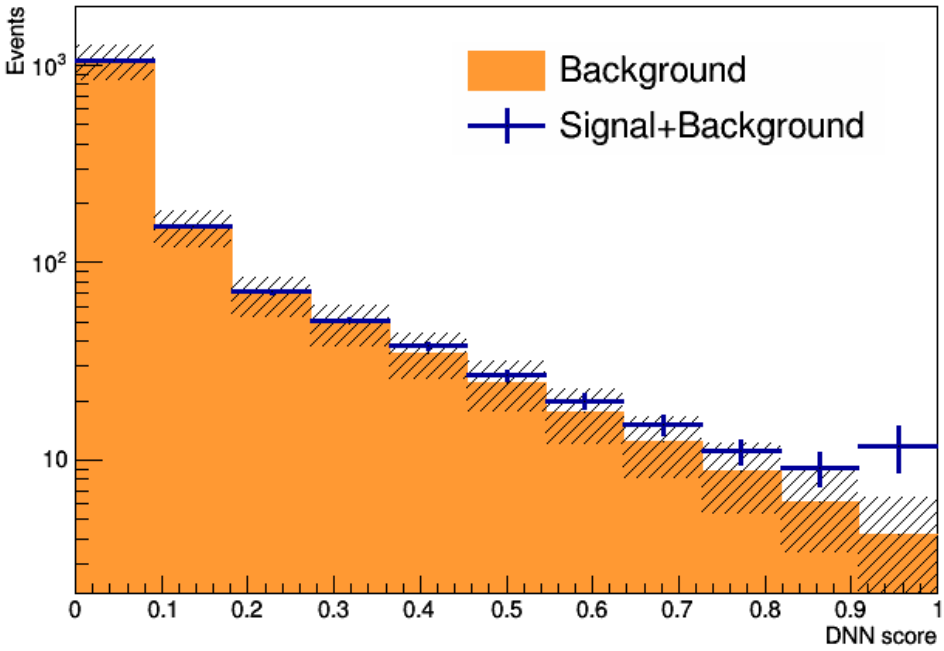


Fig. 6.8: Signal and background distributions after classification of the 8 TeV dataset corresponding to a FoM of 2.14.

7 Multivariate analysis of $H \to \tau\tau$ using the simulated 13 TeV dataset

The hyperparameters of the deep neural networks trained on the 13 TeV dataset are optimized with Spearmint. The main difference for training a neural network with either the 8 TeV or the 13 TeV dataset is the number of events in each dataset and one missing input variable, $p_{\mathrm{T}}^{\mathrm{sv}}$, in the latter. Since this variable got a high ranking when selecting the optimum subset of input variables in Sec. 6.2.2 this ranking needs to be revised. For this purpose the same procedure as in Sec. 6.2.2 is used. Additionally, the input variables are preprocessed as in Sec. 6.2.4. Figure 7.1 shows that the FoM is almost constant for any number of input variables greater or equal to 6. For fewer than 6 variables the FoM starts to decrease. Therefore, it is best to keep all variables since computation time is mostly independent from the number of input variables when using a GPU-accelerated library.

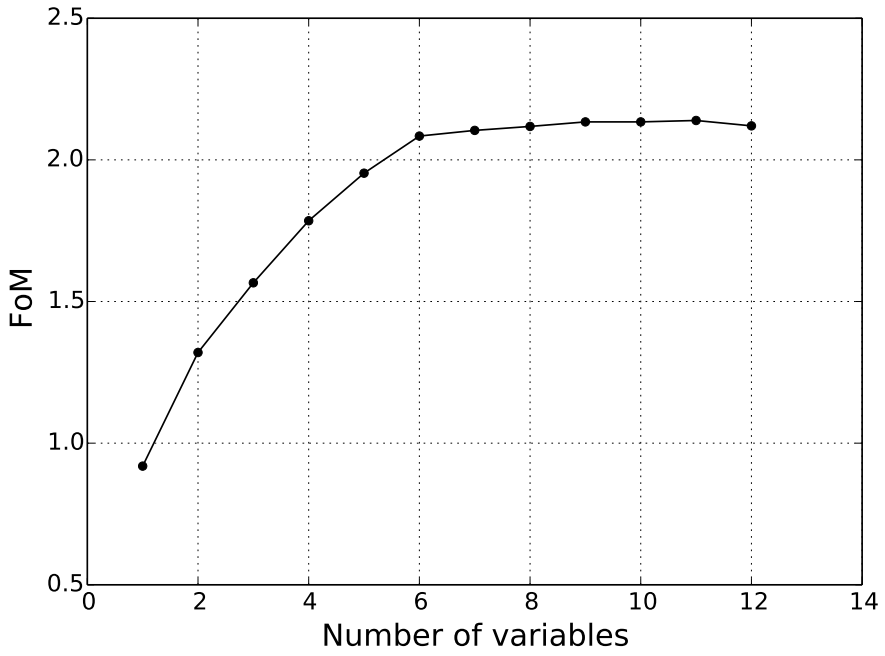


Fig. 7.1: Highest FoM as a function of the number of input variables for the 13 TeV dataset.

7.1 Performance with different activation functions

In the following section the performance of deep neural networks with different numbers of hidden layers and activation functions is discussed. The Tanh and rectified linear activation function have already been introduced in Sec. 4.2. The rectified log activation function has been designed for this thesis and is introduced in Sec. 7.1.3. The parameters which are optimized (see Tab. 7.1) are equal to the parameters in Sec. 6.2.3. The reason for using fewer nodes compared to Chap. 6 is that the performance is not improving when using more as soon as preprocessing is used on the input variables. The only significant effect of more nodes when training on the 13 TeV dataset is an increased computation time.

	Minimum	Maximum
Nodes	20	150
Learning Rate	10^{-5}	$3 \cdot 10^{-2}$
Decay Factor	$1 + 10^{-6}$	$1 + 10^{-3}$
Initial Momentum	0.5	0.8
Initial Momentum	0.8	0.995
Epochs to FM	20	200

Tab. 7.1: Parameter space of the hyperparameters used to optimize with Spearmint. The neural networks use 1 to 9 hidden layers and different activation functions.

7.1.1 Tanh activation function

Figure 7.2 shows the FoM as a function of the number of hidden layers when using the Tanh activation function. The main difference between this result and the one obtained for 8 TeV is that already 1 hidden layer leads to a high FoM. The FoM is increasing slightly when using up to 4 hidden layers but stays almost constant afterwards. The largest FoM is obtained with the maximum number of 9 hidden layers. The configuration used to obtain this result is listed in Tab. 7.2.

Hyperparameter	Value
Hidden Layers	9
Nodes	100
Learning Rate	$3 \cdot 10^{-3}$
Decay Factor	$1 + 10^{-5}$
Initial Momentum	0.66
Final Momentum	0.995
Epochs to FM	100
Figure of Merit	2.12

Tab. 7.2: Highest FoM and optimal hyperparameters when using the Tanh activation function.

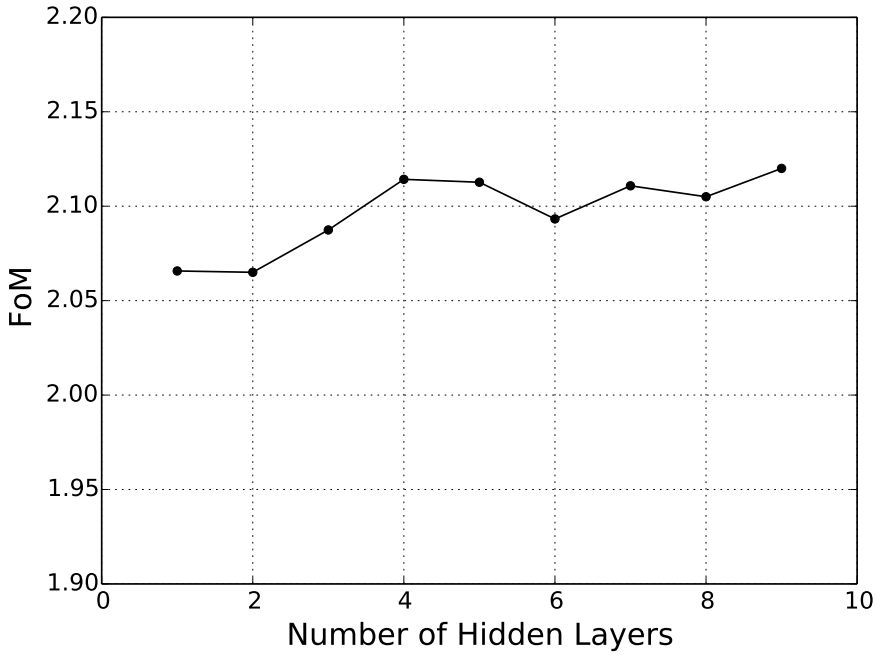


Fig. 7.2: Figure of Merit obtained with different numbers of hidden layers using the Tanh activation function.

7.1.2 Rectified linear activation function

The rectified linear activation function is 0 when the pre-activation is negative which leads to sparsity in a neural network. This means that not all neurons are active at a certain input. The advantage of a sparse representation in a deep neural network is the so-called information disentangling [62]. The objective of a deep neural network is to disentangle the components explaining the data. A dense representation is highly entangled and already small changes in the input lead to changes in the representation. When sparsity is introduced this entanglement is broken and the representation becomes more robust to small changes.

In Fig. 7.3 the results obtained with the rectified linear activation function are shown. The hyperparameter configuration leading to the highest FoM for the rectified linear activation function is listed in Tab. 7.3. The FoM is increasing when using more hidden layers and reaches the maximum when using 9 hidden layers. The lower FoM for 6 and 7 hidden layers is related to the optimization algorithm getting stuck in a local maximum.

Hyperparameter	Value
Hidden Layers	9
Nodes	85
Learning Rate	10^{-3}
Decay Factor	$1 + 3 \cdot 10^{-5}$
Initial Momentum	0.65
Final Momentum	0.995
Epochs to FM	70
Figure of Merit	2.14

Tab. 7.3: Highest FoM and optimal hyperparameters when using the rectified linear activation function.

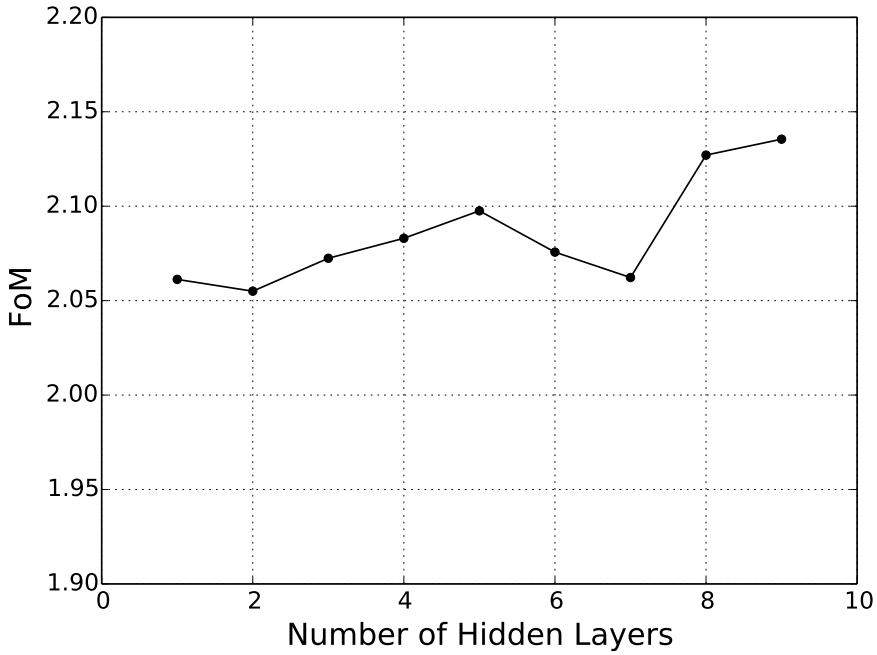


Fig. 7.3: Figure of Merit obtained with different numbers of hidden layers using the rectified linear activation function.

7.1.3 The rectified log activation function

The rectified log activation function (see Fig. 7.4)

$$a(p) = \begin{cases} \log(1+p) & \text{if } p \geq 0 \\ 0 & \text{else} \end{cases}, \tag{7.1}$$

is intended to introduce sparsity similar to the rectified linear activation function in the deep neural network but also a non-linearity for a positive pre-activation. The rectified log has no upper bound and is increasing more slowly than any polynomial function towards positive infinity. The rectified log achieved the highest FoM of all three tested activation functions. However, the difference between the three results is not statistically significant. This value was not reached with the maximum number of 9 hidden layers but already with 7 layers as shown in Fig. 7.5. The hyparameters used to obtain this FoM are listed in Tab. 7.4. The distribution of the signal and background after classification is shown in Fig. 7.6. The b_{reg} term in eq. 5.1 is set to $b_{reg} \approx 0$ for the computation of the final FoM.

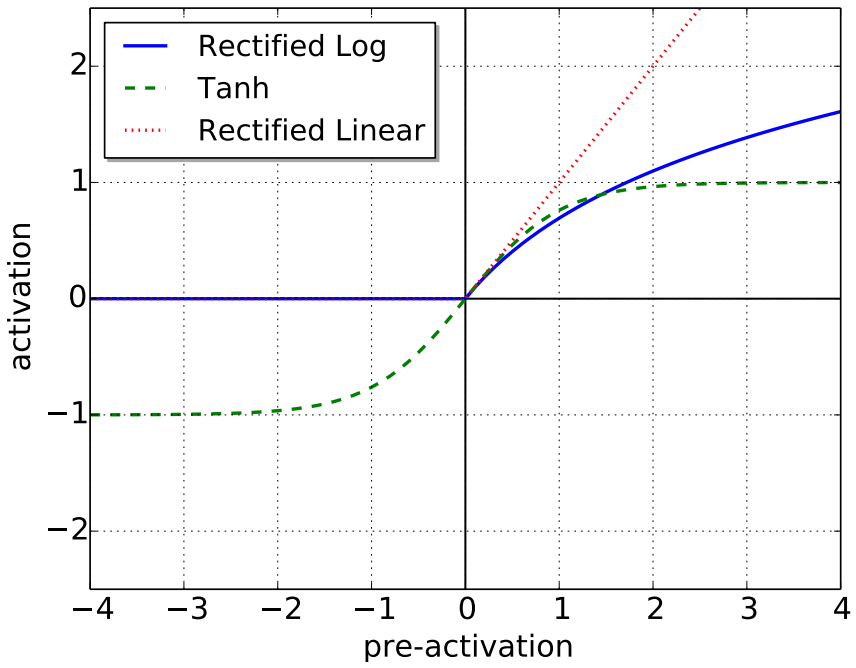


Fig. 7.4: Rectified log activation function compared to the Tanh and the rectified linear activation function.

Hyperparameter	Value
Hidden Layers	7
Nodes	108
Learning Rate	$3 \cdot 10^{-3}$
Decay Factor	$1 + 10^{-5}$
Initial Momentum	0.66
Final Momentum (FM)	0.995
Epochs to FM	112
Figure of Merit	2.15

Tab. 7.4: Highest FoM and optimal hyperparameters when using the rectified log activation function.

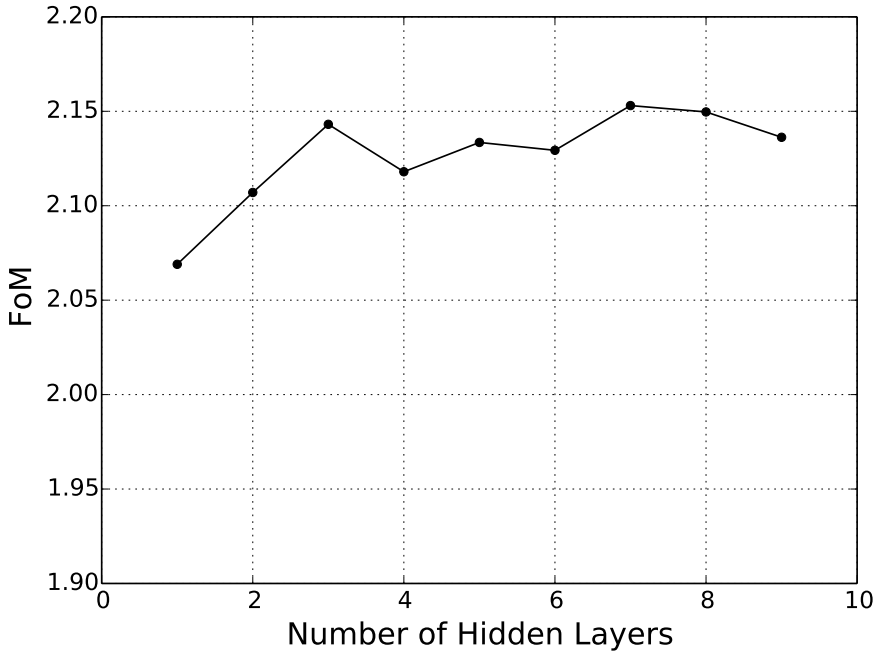


Fig. 7.5: Figure of Merit obtained with different numbers of hidden layers using the rectified log activation function.

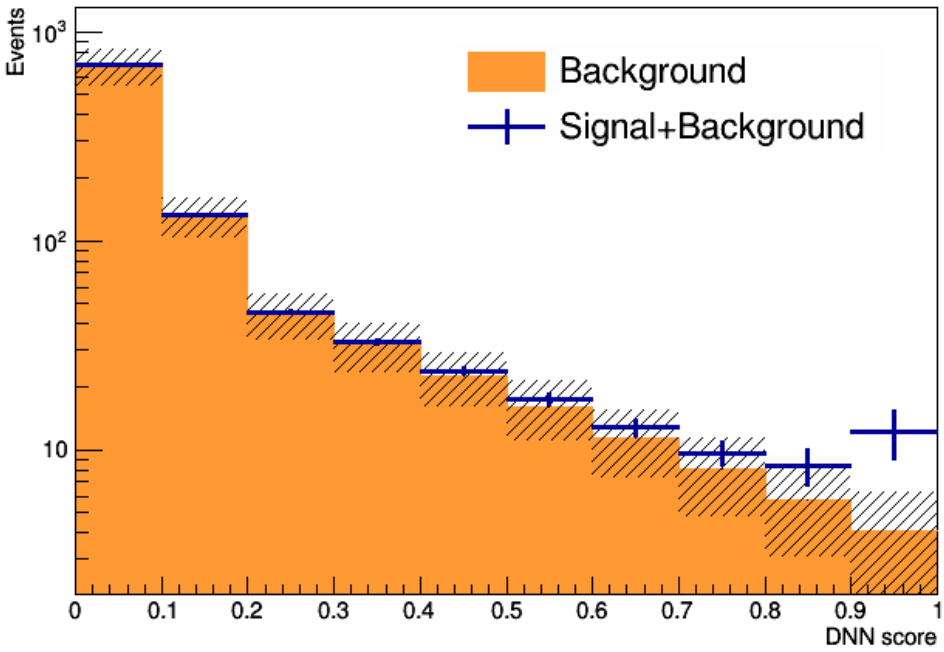


Fig. 7.6: Signal and background distributions after classification of the 13 TeV dataset. The deep neural network used the rectified log activation function and achieved a FoM of 2.15.

7.2 Influence of dataset size on network performance

In Sec. 6.2.3 the FoM did not increase further for 8 hidden layers which suggested that the 8 TeV dataset is too small. The 13 TeV dataset is much larger (see Tab. 5.3), and it is therefore possible to investigate the influence of the dataset size on the performance of the deep neural network. For this purpose the configuration in Tab. 7.4 is used. This network is trained on subsets of the 13 TeV dataset with different sizes. Training is performed several times for each subset with differently initialized weights. The size is increased step by step up to the total number of events in the dataset. In Fig. 7.7 the FoM for different numbers of events in the training is shown. The points represent the mean value of the results obtained with differently initialized weights. The error bars correspond to the standard deviation. The FoM shows no clear dependency on the training set size. One possible explanation is that the deep neural network is not able to perform any better even when the dataset size is increased. Another, more likely, explanation is that the size of the 13 TeV dataset, like the 8 TeV dataset, is too small to achieve optimum performance. This conclusion is based on a lecture of Peter Norvig [63] who stated that deep neural networks typically need much more training events than provided in the 13 TeV dataset to start improving in performance. Below this threshold the performance of a deep neural networks is independent of the dataset size and shows no "discrete" improvement when using larger datasets as one would assume.

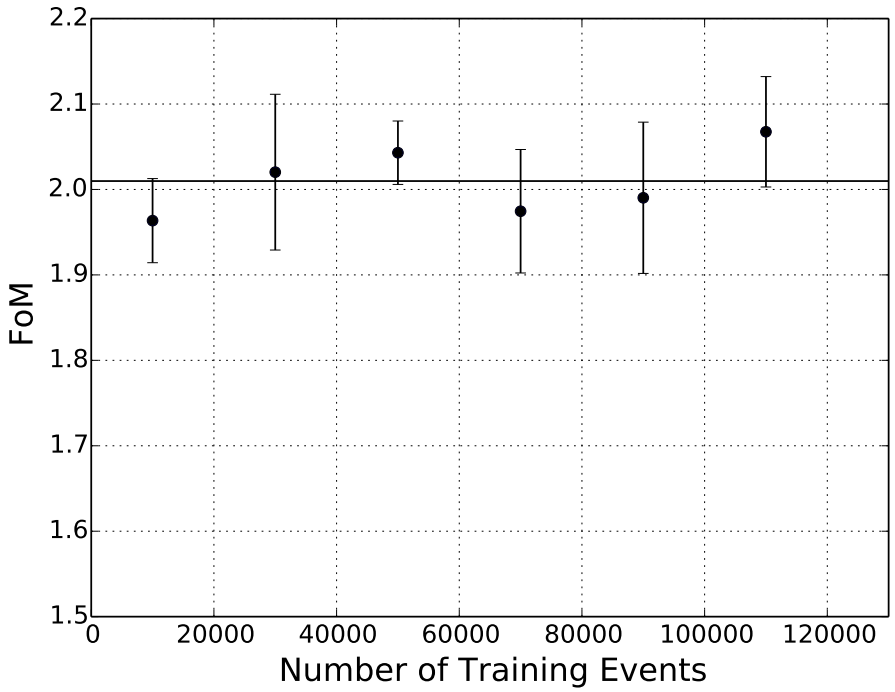


Fig. 7.7: Performance of a deep neural network when trained with different dataset sizes.

8 Summary and conclusions

Deep Learning is the current state-of-the-art machine learning approach in speech and image recognition. It is therefore reasonable to assume that deep learning can improve the classification of data in high energy physics as well. For this purpose a deep neural network is tested in order to classify signal and background events given in a simulated $H \to \tau\tau$ dataset for 8 TeV corresponding to an integrated luminosity of $\mathcal{L} = 19.7\,\mathrm{fb}^{-1}$ and a simulated dataset for 13 TeV corresponding to an integrated luminosity of $\mathcal{L} = 10\,\mathrm{fb}^{-1}$.

The 8 TeV dataset is mainly used to investigate different optimization approaches. At first grid search is used to optimize the hyperparameters for network architecture and learning algorithm. The main struggle with grid search is the missing computation power even though a custom built workstation is used which used a mid-range GPU to accelerate computations. Therefore, a Bayesian optimization algorithm is chosen to optimize the hyperparameters leading to better results in a shorter time. Using this optimization algorithm the next step is to find the best subset of input variables. Finally, preprocessing is used on the input variables resulting in the best FoM for the 8 TeV dataset (see Tab. 8.1).

The insight gained during optimization of the 8 TeV dataset is then used on the 13 TeV dataset. Additionally three different activation functions are tested on this dataset. The last step after optimizing all hyperparameters and specifying the best activation function is to determine the influence of the dataset size on the deep neural network training.

Parallel to this thesis two other machine learning approaches were tested on the same datasets. These were Boosted Decision Trees (BDTs) [26–28] and NeuroBayes [64, 65] which are available in the multivariate analysis implementation (TMVA [66]) of ROOT [67]. Also a cut-based approach is tested to which all three machine learning approaches are compared to.

The best FoM obtained with deep neural networks, NeuroBayes, BDTs and cut-based are listed for both datasets in Tab. 8.1. A comparison shows that NeuroBayes and BDTs perform much better than the remaining approaches while deep neural networks only perform equal to slightly better than the cut-based approach.

Approach	cut-based	Deep Learning	NeuroBayes	BDT
8 TeV	2.16 [68]	2.14	2.96 [69]	3.54 [68]
13 TeV	1.94 [68]	2.15	3.12 [70]	3.81 [68]

Tab. 8.1: Highest FoM for different analysis techniques, assuming an integrated luminosity of $\mathcal{L} = 19.7\,\text{fb}^{-1}$ for the 8 TeV dataset and an integrated luminosity of $\mathcal{L} = 10\,\text{fb}^{-1}$ for the 13 TeV dataset. The b_{reg} term in eq. 5.1 is set to $b_{reg} \approx 0$ for the calculation of the final FoM.

While deep neural networks are the cutting-edge technology in the field of speech and image recognition it is not obvious that this applies to the data analysis performed in this thesis as well. However, other studies [44, 71] already showed that deep neural networks are able to outperform shallow neural networks significantly. The datasets used in these studies are each about 100 – 1000 times larger than the simulated 13 TeV dataset used in this thesis. Furthermore, a high-end GPU especially designed for data analysis is used which made it possible to use such large datasets for training in the first place. In comparison, the shallow neural networks trained on the 13 TeV dataset in this thesis perform almost as well as a deep neural network with many hidden layers. A possible explanation why more hidden layers do not lead to a significantly better performance is based on the vanishing gradient problem and therefore indirectly on the size of the training dataset. Hochreiter [43] showed in 1991 that the back-propagated error decays exponentially from layer to layer. Therefore, to train all layers of a deep neural network sufficiently one needs to train the network long enough. A large dataset is therefore necessary when training deep neural networks since it leads automatically to a large number of training updates while preventing overtraining. Furthermore, Peter Norvig stated during the Vienna Gödel Lecture 2015 [63] that deep neural networks require a very large dataset in order to get an actual improvement in performance when using more hidden layers compared to shallow neural networks. The performance on sub-samples of the 13 TeV dataset showed no dependency on the size of this sub-samples. This suggests that the 13 TeV dataset size is well below the threshold above which more hidden layers lead to an improvement in performance.

The final conclusion of this thesis is that the effort to optimize a deep neural network and the resulting performance on the available datasets is not comparable to the high "out-of-the-box" performance delivered by BDTs. Without a much larger dataset and the necessary computation power to handle this amount of data the true prowess of deep neural networks compared to other approaches cannot be explored.

Appendices

A Acronyms

AMS Approximate Median Significance

APD Avalanche Photodiodes

BDT Boosted Decision Tree

CPU Central Processing Unit

CMS Compact Muon Solenoid

CSC Cathode Strip Chamber

DNN Deep Neural Network

DT Drift Tube

ECAL Electromagnetic Calorimeter

eV electron Volt

FM Final Momentum

FoM Figure of Merit

GPU Graphics Processing Unit

ggF gluon-gluon-Fusion

GUT Grand Unified Theory

HB Hadronic Barrel Calorimeter

HCAL Hadronic Calorimeter

HE Hadronic Endcap Calorimeter

HF Hadronic Forward Calorimeter

HO Hadronic Outer Barrel Calorimeter

LHC Large Hadron Collider

ML Machine Learning

MLP Multilayer Perceptron

MVA Multivariate Analysis

RPC Resistive Plate Chamber

SGD Stochastic Gradient Descent

SM Standard Model of particle physics

Tanh Hyperbolic tangent

TEC Tracker Endcap

TIB Tracker Inner Barrel

TID Tracker Inner Discs

TOB Tracker Outer Barrel

VBF Vector Boson Fusion

VPT Vacuum Phototriodes

B Framework setup

In the following section the required python packages and necessary settings in order to run the code available at the following link is summarized:

- `https://github.com/MarkusSpanring/hepDnn.git`

The packages where installed on `Ubuntu Linux 14.04 LTS 64-Bit` with superuser rights. The instructions should be followed step by step to get a working framework. Alternatively one can follow the instructions given at:

- `http://deeplearning.net/software/pylearn2`
- `https://github.com/HIPS/Spearmint`

Step 1) Installing Theano

Command	Version
`apt-get install python-dev`	2.7.5
`apt-get install python-numpy`	1.8.1
`apt-get install python-scipy`	0.13.3
`apt-get install python-pip`	1.5.4
`apt-get install python-git`	1.9.1
`apt-get install python-nose`	1.3.1
`apt-get install g++`	4.8.2
`apt-get install libopenblas-dev`	0.2.8

The package `python-pip` is necessary to use `pip install`

`pip install theano`	0.6.0

Step 2) Installing CUDA

The installation of CUDA is necessary to enable GPU capabilities of theano. If only the CPU is used this step is not needed.

Command	Version
`apt-get install nvidia-current`	1.8.1
`apt-get install nvidia-cuda-toolkit`	5.5.22
`apt-get update`	

IMPORTANT!! After downloading the current Nvidia driver with `apt`
it is necessary to manually load the driver in Ubuntu and reboot afterwards.
Otherwise CUDA can not be installed successfully.

To enable theano for GPU usage specific theano-flags need to be set which is done in a `.theanorc` file one has to create in the home directory. This file needs the following content:

```
[global]
floatX=float32
device=gpu
```

More information on the usage of theano-flags can be found at:

- `http://deeplearning.net/software/theano/library/config.html`

Step 3) Installing Pylearn2

Command	Version
`apt-get install cython`	0.20.1
`pip install scikit-learn`	0.14.1
`apt-get install python-matplotlib`	1.3.1
`apt-get install python-yaml`	3.10
`git clone git://github.com/lisa-lab/pylearn2.git`	0.1
`python ~/pylearn2/setup.py develop`	

Step 4) Installing ROOT

Command	Version
`apt-get install root-system`	5.34
`apt-get install libroot-bindings-python-dev`	

Step 5) Installing Spearmint

Command	Version
`git clone https://github.com/HIPS/Spearmint.git`	0.1
`pip install -e  /Spearmint`	
`pip install pymongo`	3.0.2

For further information on installing and setting up MongoDB see:

- `https://www.mongodb.org`

Step 6) Setting up the environment

The most important paths are saved as environment variable in order to make the installed python code more flexible.

Environment variable name	Description
`PYLEARN2_DATA_PATH`	Path to training and test dataset
`DNN_PATH`	`~/hepDnn`
`PYTHONPATH`	`/usr/lib/x86_64-linux-gnu/root5.34`

If theano is used in combination with a Nvidia GPU superuser rights are necessary. However, when using `sudo python` to run python code the user environment variables are no longer available. A quick way around is to add these variables in the `sudoers` file.

Command	Description
`sudo visudo`	Opens the sudoers file

Lines to add in sudoers
`Defaults env_keep+=PYLEARN2_DATA_PATH`
`Defaults env_keep+=DNN_PATH`
`Defaults env_keep+=PYTHONPATH`

Saving file with: `STRG+X`

Bibliography

1. Beringer, J. *et al.* Review of Particle Physics. *Phys. Rev. D* **86,** 010001 (2012).
2. 't Hooft, G. & Veltman, M. Regularization and renormalization of gauge fields. *Nuclear Physics B* **44,** 189–213 (1972).
3. Narison, S. *QCD as a Theory of Hadrons* (Cambridge University Press, 2004).
4. Perl, M. L. *et al.* Evidence for anomalous lepton production in e^+e^- annihilation. *Physical Review Letters* **35,** 1489–1492 (Dez. 1975).
5. Cottingham, W. & Greenwood, D. *An Introduction to the Standard Model of Particle Physics* (Cambridge University Press, 2007).
6. Misner, Charles W. and Thorne, K.S. and Wheeler, J.A. *Gravitation* (W.H. Freeman und Company, 1974).
7. Maxwell, J. C. A dynamical theory of the electromagnetic field. *Philosophical Transactions of the Royal Society of London* **155,** 459–513 (1865).
8. Einstein, A. On the Electrodynamics of Moving Bodies. *Annalen der Physik* **17** (1905).
9. Einstein, A. Über einen die Erzeugung und Verwandlung des Lichtes betreffenden heuristischen Gesichtspunkt. *Ann. Phys.* **322,** 132–148 (1905).
10. Dirac, P. A. M. The Quantum Theory of the Electron. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* **117,** 610–624 (1928).
11. Glashow, S. Partial Symmetries of Weak Interactions. *Nucl.Phys.* **22,** 579–588 (1961).
12. Weinberg, S. A Model of Leptons. *Phys.Rev.Lett.* **19,** 1264–1266 (1967).
13. Higgs, P. W. Broken symmetries and the masses of gauge bosons. *Phys. Rev. Lett.* **13,** 508–509 (1964).
14. Fanti, V. *et al.* A New measurement of direct CP violation in two pion decays of the neutral kaon. *Phys.Lett.* **B465,** 335–348 (1999).
15. Goldstone, J., Salam, A. & Weinberg, S. Broken Symmetries. *Phys. Rev.* **127,** 965–970 (3 Aug. 1962).
16. CMS Collaboration. Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC. *Phys.Lett.* **B716,** 30–61 (2012).
17. CMS Collaboration. Observation of a new boson with mass near 125 GeV in pp collisions at $\sqrt{s}$=7 and 8 TeV. English. *Journal of High Energy Physics* **2013** (2013).

18. ATLAS Collaboration. Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC. *Phys.Lett.* **B716,** 1–29 (2012).

19. CMS Collaboration. Study of the Mass and Spin-Parity of the Higgs Boson Candidate Via Its Decays to Z Boson Pairs. *Phys.Rev.Lett.* **110,** 081803 (2013).

20. Dittmaier, S. *et al.* Handbook of LHC Higgs Cross Sections: 1. Inclusive Observables. doi:10.5170/CERN-2011-002 (2011).

21. Heinemeyer, S. *et al. Handbook of LHC Higgs Cross Sections: 3. Higgs Properties: Report of the LHC Higgs Cross Section Working Group* Techn. Ber. arXiv:1307.1347. CERN-2013-004 (Geneva, 2013).

22. Yukawa, H. On the Interaction of Elementary Particles. I. *Proceedings of the Physico-Mathematical Society of Japan. 3rd Series* **17,** 48–57 (1935).

23. ATLAS collaboration. Evidence for Higgs Boson Decays to the $\tau^+\tau^-$ Final State with the ATLAS Detector (2013).

24. CMS Collaboration. Evidence for the 125 GeV Higgs boson decaying to a pair of τ leptons. *JHEP* **1405,** 104 (2014).

25. ATLAS Collaboration. The ATLAS Experiment at the CERN Large Hadron Collider. *JINST* **3,** S08003 (2008).

26. Breiman, L., Friedman, J. H., Olshen, R. A. & Stone, C. J. *Classification and Regression Trees* (Wadsworth Publishing Company, Belmont, California, U.S.A., 1984).

27. Freund, Y. & Schapire, R. E. *A Decision-theoretic Generalization of On-line Learning and an Application to Boosting* in *Proceedings of the Second European Conference on Computational Learning Theory* (Springer-Verlag, London, UK, UK, 1995), 23–37.

28. Friedman, J. H. Stochastic Gradient Boosting. *Comput. Stat. Data Anal.* **38,** 367–378 (Feb. 2002).

29. Arnison, G. *et al.* Experimental Observation of Isolated Large Transverse Energy Electrons with Associated Missing Energy at $\sqrt{s}$ = 540-GeV. *Phys.Lett.* **B122,** 103–116 (1983).

30. Baur, G. *et al.* Production of anti-hydrogen in relativistic collisions. *Nucl. Instrum. Meth.* **A391,** 201–204 (1997).

31. Lefevre, C. *LHC: the guide* Feb. 2009.

32. CMS Collaboration. *CMS Physics: Technical Design Report Volume 1: Detector Performance and Software* (CERN, Geneva, 2006).

33. CMS Collaboration. The CMS experiment at the CERN LHC. *JINST* **3,** S08004 (2008).

34. CMS Collaboration. *CMS detector overview* retrieved: March. 2015. <http://cmsinfo.web.cern.ch/cmsinfo/Detector/FullDetector/index.html>.

35. Glorot, X., Bordes, A. & Bengio, Y. *Deep Sparse Rectifier Neural Networks* in. **15** (Journal of Machine Learning Research - Workshop und Conference Proceedings, 2011), 315–323.

36. Minsky, M. L. & Papert, S. A. *Perceptrons: An Introduction to Computational Geometry* (The MIT Press, 1990).

37. Russell, S. J. & Norvig, P. *Artificial Intelligence: A Modern Approach* 2. Aufl. (Pearson Education, 2003).

38. Hornik, K., Stinchcombe, M. & White, H. Multilayer Feedforward Networks Are Universal Approximators. *Neural Networks* **2,** 359–366 (Juli 1989).

39. Bengio, Y. *Learning Algorithms* in (2010). <`http://www.iro.umontreal.ca/~pift6266/H10/notes/contents.html`>.

40. Kullback, S. & Leibler, R. A. On Information and Sufficiency. *Ann. Math. Statist.* **22,** 79–86 (März 1951).

41. Hinton, G. E. in *Neural Networks: Tricks of the Trade (2nd ed.)* 599–619 (Springer, 2012).

42. Rumelhart, D. E., Hinton, G. E. & Williams, R. J. in *Neurocomputing: Foundations of Research* 696–699 (MIT Press, Cambridge, MA, USA, 1988).

43. Hochreiter, S. *Untersuchungen zu dynamischen neuronalen Netzen. Diploma thesis, Institut für Informatik, Lehrstuhl Prof. Brauer, Technische Universität München* 1991.

44. Baldi, P., Sadowski, P. & Whiteson, D. Searching for Exotic Particles in High-Energy Physics with Deep Learning. arXiv: `1402.4735 [hep-ph]` (2014).

45. Goodfellow, I. J. *et al.* Pylearn2: a machine learning research library. *ArXiv e-prints.* arXiv: `1308.4214 [stat.ML]` (Aug. 2013).

46. Bergstra, J. *et al. Theano: a CPU and GPU Math Expression Compiler* in *Proceedings of the Python for Scientific Computing Conference (SciPy)* Oral Presentation (Austin, TX, Juni 2010).

47. Bastien, F. *et al. Theano: new features and speed improvements* Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop. 2012.

48. Snoek, J., Larochelle, H. & Adams, R. P. *Practical Bayesian Optimization of Machine Learning Algorithms* in *Advances in Neural Information Processing Systems 25* (Dez. 2012).

49. Rasmussen, C. E. & Williams, C. K. I. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)* (The MIT Press, 2005).

50. Nason, P. A New method for combining NLO QCD with shower Monte Carlo algorithms. *JHEP* **11,** 040 (2004).

51. Frixione, S., Nason, P. & Oleari, C. Matching NLO QCD computations with Parton Shower simulations: the POWHEG method. *JHEP* **11,** 070 (2007).

52. Alioli, S., Nason, P., Oleari, C. & Re, E. A general framework for implementing NLO calculations in shower Monte Carlo programs: the POWHEG BOX. *JHEP* **06,** 043 (2010).

53. Alioli, S., Hamilton, K., Nason, P., Oleari, C. & Re, E. Jet pair production in POWHEG. *JHEP* **04,** 081 (2011).

54. Alioli, S., Nason, P., Oleari, C. & Re, E. NLO Higgs boson production via gluon fusion matched with shower in POWHEG. *JHEP* **04,** 002 (2009).

55. Sjöstrand, T., Mrenna, S. & Skands, P. PYTHIA 6.4 physics and manual. *Journal of High Energy Physics* **2006,** 026 (2006).

56. Alwall, J., Herquet, M., Maltoni, F., Mattelaer, O. & Stelzer, T. MadGraph 5 : Going Beyond. *JHEP* **06,** 128 (2011).

57. Sjöstrand, T. *et al.* An Introduction to PYTHIA 8.2. *Comput. Phys. Commun.* **191,** 159–177 (2015).

58. Agostinelli, S. *et al.* GEANT4: A Simulation toolkit. *Nucl. Instrum. Meth.* **A506,** 250–303 (2003).

59. ATLAS collaboration. Evidence for Higgs boson Yukawa couplings in the $H \to \tau\tau$ decay mode with the ATLAS detector (2014).

60. Brandstetter, J. *Private conversation with Johannes Brandstetter* HEPHY Vienna. 2015.

61. Kotsiantis, S. B. & et al. *Data Preprocessing for Supervised Learning* 2006.

62. Bengio, Y. Learning deep architectures for AI. *Foundations and Trends in Machine Learning* **2.** Also published as a book. Now Publishers, 2009., 1–127 (2009).

63. Norvig, P. *How Computers learn* in *Vienna Gödel Lecture* (2015). <`http://www.informatik.tuwien.ac.at/vienna-goedel-lectures/2015`>.

64. Feindt, M. A Neural Bayesian Estimator for Conditional Probability Densities. arXiv: `physics/0402093 [physics.data-an]` (2004).

65. Feindt, M. & Kerzel, U. The NeuroBayes neural network package. *Nucl. Instrum. Meth.* **A559,** 190–194 (2006).

66. Hoecker, A. *et al.* TMVA: Toolkit for Multivariate Data Analysis. *PoS* **ACAT,** 040 (2007).

67. Brun, R. & Rademakers, F. ROOT: An object oriented data analysis framework. *Nucl. Instrum. Meth.* **A389,** 81–86 (1997).

68. Schamböck, V. *Private conversation with Verena Schamböck* HEPHY Vienna. 2015.

69. Kloibhofer, S. *Potential of the analysis of Higgs boson decays to two tau leptons with NeuroBayes with the CMS experiment* Vienna University of Technology. 2015.

70. Brondolin, E. *Private conversation with Erica Brondolin* HEPHY Vienna. 2015.

71. Baldi, P., Sadowski, P. & Whiteson, D. Enhanced Higgs to $\tau^+\tau^-$ Searches with Deep Learning. arXiv: `1410.3469 [hep-ph]` (2014).